ssm环境配置文档

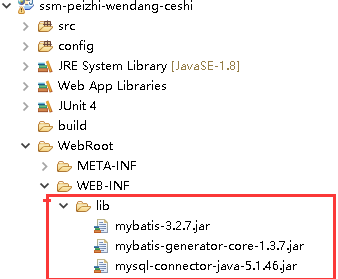
1. 配置逆向工程自动生成实体类、mapper结构和sql映射

第一步：导包（3个）到lib文件夹

mybatis-3.2.7.jar

mybatis-generator-core-1.3.7.jar

mysql-connector-java-5.1.46.jar



第二步：创新自动生成的.xml文件 mbg.xml

1.1配置数据源  
 <jdbcConnection driverClass=*"com.mysql.jdbc.Driver"*

connectionURL=*"jdbc:mysql://127.0.0.1:3306/mybatis01?useUnicode=true&amp;characterEncoding=utf8&amp;useSSL=true"*

userId=*"root"*

password=*"1997"*>

</jdbcConnection>

1.2java类型处理器

<!-- 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** >

<!--

true：使用BigDecimal对应DECIMAL和 NUMERIC数据类型

false：默认,

scale>0;length>18：使用BigDecimal;

scale=0;length[10,18]：使用Long；

scale=0;length[5,9]：使用Integer；

scale=0;length<5：使用Short；

-->  
 <**property** *name***="forceBigDecimals"** *value***="false"** />  
</**javaTypeResolver**>

1.3根据表生成实体类

<!-- 根据表生成实体类 -->

<**javaModelGenerator** *targetPackage***="com.yb.entity"** *targetProject***=".\src"**>  
 <!-- 是否子包 -->  
 <**property** *name***="enableSubPackages"** *value***="true"** />  
 <!-- 是否去空格 -->

<**property** *name***="trimStrings"** *value***="true"** />

</**javaModelGenerator**>

1.4生成mapper映射

<!-- 根据表生成mapper映射 -->  
<**sqlMapGenerator** *targetPackage***="mapper"** *targetProject***=".\config"**>  
 <**property** *name***="enableSubPackages"** *value***="true"** />  
</**sqlMapGenerator**>

1.5生成mapper接口

<**javaClientGenerator** *type***="XMLMAPPER"** *targetPackage***="com.yb.mapper"** *targetProject***=".\src"**>  
 <**property** *name***="enableSubPackages"** *value***="true"** />  
</**javaClientGenerator**>

1.6数据的表名和实体类名

<**table** *tableName***="edoc\_entry"** *domainObjectName***="EdocEntry"** >  
   
</**table**>  
<**table** *tableName***="edoc\_category"** *domainObjectName***="EdocCategory"** >  
  
</**table**>

整体代码：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE generatorConfiguration

PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"

"http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd">

<generatorConfiguration>

<!--

targetRuntime：MyBatis3Simple 只生产简单的增删改查

MyBatis3：还包括了支持高级查询（豪华版）

-->

<context id=*"DB2Tables"* targetRuntime=*"MyBatis3"*>

<!-- 序列化实体类 -->

<plugin type=*"org.mybatis.generator.plugins.SerializablePlugin"*/>

<!-- 自动生产toString -->

<plugin type=*"org.mybatis.generator.plugins.ToStringPlugin"*/>

<!-- 自动生成 equals() hashcode() -->

<plugin type=*"org.mybatis.generator.plugins.EqualsHashCodePlugin"*/>

<jdbcConnection driverClass=*"com.mysql.jdbc.Driver"*

connectionURL=*"jdbc:mysql://127.0.0.1:3306/mybatis01?useUnicode=true&amp;characterEncoding=utf8&amp;useSSL=true"*

userId=*"root"*

password=*"1997"*>

</jdbcConnection>

<!-- java类型处理器 -->

<!-- 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 >

<!--

true：使用BigDecimal对应DECIMAL和 NUMERIC数据类型

false：默认,

scale>0;length>18：使用BigDecimal;

scale=0;length[10,18]：使用Long；

scale=0;length[5,9]：使用Integer；

scale=0;length<5：使用Short；

-->

<property name=*"forceBigDecimals"* value=*"false"* />

</javaTypeResolver>

<!-- 1、根据表生成实体类： -->

<javaModelGenerator targetPackage=*"com.fzq.entity"* targetProject=*"./src"*>

<!-- 不要创建子包 -->

<property name=*"enableSubPackages"* value=*"true"* />

<!-- 去空格 -->

<property name=*"trimStrings"* value=*"true"* />

</javaModelGenerator>

<!-- 2、根据表生成mapper映射： -->

<sqlMapGenerator targetPackage=*"mapper"* targetProject=*"./config"*>

<!-- 不要创建子包 -->

<property name=*"enableSubPackages"* value=*"true"* />

</sqlMapGenerator>

<!-- 3、根据表生成mapper接口： -->

<javaClientGenerator type=*"XMLMAPPER"* targetPackage=*"com.fzq.mapper"* targetProject=*"./src"*>

<!-- 不要创建子包 -->

<property name=*"enableSubPackages"* value=*"true"* />

</javaClientGenerator>

<!-- 数据的表名和实体类名 -->

<table tableName=*"edoc\_category"* domainObjectName=*"EdocCategory"* />

<table tableName=*"edoc\_entry"* domainObjectName=*"EdocEntry"* />

<!-- <generatedKey column="ID" sqlStatement="mysql" identity="true" /> -->

</context>

</generatorConfiguration>

第三步：创建实现类实现代码GeneratorTest.java

**public** **class** GeneratorTest {

@Test

**public** **void** testMBG() **throws** Exception{

List<String> warnings = **new** ArrayList<String>();

**boolean** overwrite = **true**;

File configFile = **new** File("mbg.xml");

ConfigurationParser cp = **new** ConfigurationParser(warnings);

Configuration config = cp.parseConfiguration(configFile);

DefaultShellCallback callback = **new** DefaultShellCallback(overwrite);

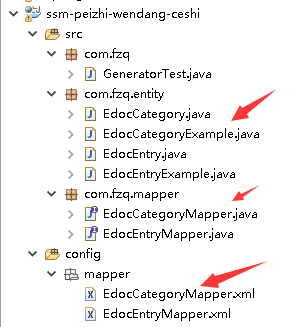
MyBatisGenerator myBatisGenerator = **new** MyBatisGenerator(config, callback, warnings);

myBatisGenerator.generate(**null**);

}

}

刷新一下就会出现：



二、sping整合mybatis

第一步：创建sping-datasources

2.1数据源

<bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"* destroy-method=*"close"*>

<property name=*"driverClassName"* value=*"${driver}"* />

<property name=*"url"* value=*"${url}"* />

<property name=*"username"* value=*"${uname}"* />

<property name=*"password"* value=*"${password}"* />

</bean>

2.2 动态代理sqlsessionFactory以及加载分页插件

<bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>

<!-- 别名 -->

<property name=*"typeAliasesPackage"* value=*"com.fzq.model"*></property>

<!-- mapper XML映射 -->

<property name=*"mapperLocations"* value=*"classpath\*:mapper/\*Mapper.xml"*></property>

<!-- 数据源 -->

<property name=*"dataSource"* ref=*"dataSource"*></property>

<!-- 分页插件 -->

<property name=*"plugins"*>

<array>

<bean class=*"com.github.pagehelper.PageInterceptor"*>

<property name=*"properties"*>

<!--使用下面的方式配置参数，一行配置一个 -->

<value>

helperDialect=mysql

</value>

</property>

</bean>

</array>

</property>

</bean>

2.3动态代理mapper接口

<bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>

<!-- 扫描mapper接口的包 -->

<property name=*"basePackage"* value=*"com.fzq.mapper"*></property>

</bean>

<bean id=*"txManager"* class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>

<property name=*"dataSource"* ref=*"dataSource"*/>

</bean>

<bean id=*"sharedTransactionTemplate"*

class=*"org.springframework.transaction.support.TransactionTemplate"*>

<constructor-arg name=*"transactionManager"* ref=*"txManager"*/>

</bean>

</beans>

2.4创建jdbc.properties文件

driver=com.mysql.jdbc.Driver

url=jdbc:mysql://127.0.0.1:3306/mybatis01?useUnicode=true&characterEncoding=utf8&useSSL=true

uname=root

password=1997

整体代码：

<?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:p=*"http://www.springframework.org/schema/p"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*https://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*https://www.springframework.org/schema/context/spring-co ntext.xsd*

*"*>

<!-- 1，加载数据信息属性文件 -->

<context:property-placeholder location=*"classpath:jdbc.properties"*/>

<!-- 2,数据源：数据库的相关信息 :官网有 bdcp数据源，是数据源的一种 -->

<bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"* destroy-method=*"close"*>

<property name=*"driverClassName"* value=*"${driver}"* />

<property name=*"url"* value=*"${url}"* />

<property name=*"username"* value=*"${uname}"* />

<property name=*"password"* value=*"${password}"* />

</bean>

<!-- 3、sqlSessionFactory 对象 -->

<bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>

<!-- 别名 -->

<property name=*"typeAliasesPackage"* value=*"com.fzq.model"*></property>

<!-- mapper XML映射 -->

<property name=*"mapperLocations"* value=*"classpath\*:mapper/\*Mapper.xml"*></property>

<!-- 数据源 -->

<property name=*"dataSource"* ref=*"dataSource"*></property>

<!-- 分页插件 -->

<property name=*"plugins"*>

<array>

<bean class=*"com.github.pagehelper.PageInterceptor"*>

<property name=*"properties"*>

<!--使用下面的方式配置参数，一行配置一个 -->

<value>

helperDialect=mysql

</value>

</property>

</bean>

</array>

</property>

</bean>

<!-- 4、Mapper接口的代理对象 -->

<bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>

<!-- 扫描mapper接口的包 -->

<property name=*"basePackage"* value=*"com.fzq.mapper"*></property>

</bean>

<bean id=*"txManager"* class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>

<property name=*"dataSource"* ref=*"dataSource"*/>

</bean>

<bean id=*"sharedTransactionTemplate"*

class=*"org.springframework.transaction.support.TransactionTemplate"*>

<constructor-arg name=*"transactionManager"* ref=*"txManager"*/>

</bean>

</beans>

第二步：创建context.xml引入sping-datasources

<?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:p=*"http://www.springframework.org/schema/p"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*https://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*https://www.springframework.org/schema/context/spring-context.xsd*

*"*>

<!-- 1、注解扫描 -->

<context:component-scan base-package=*"com.fzq.\*\*"*>

<!-- 排除@Controller注解 -->

<context:exclude-filter type=*"annotation"* expression=*"org.springframework.stereotype.Controller"*/>

</context:component-scan>

<!-- 2、datasource:mybatis -->

<import resource=*"spring-datasource.xml"*/>

<!-- 3、事务管理器 -->

</beans>

导包：

spring-beans-5.0.14.RELEASE.jar

spring-test-5.0.14.RELEASE.jar

1. 搭建springmvc并整合spring

第一步：创建spring-web.xml

3.1、注解扫描

<**context:component-scan** *base-package***="com.yb.web"**/>

3.2、映射器

<**bean** *class***="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"**/>

3.3、适配器以及json类转换器

<**bean** *class***="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"**/>

3.4、视图解析器

<**bean** *class***="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  
 <!-- 解析jstl标签 -->  
 <**property** *name***="viewClass"** *value***="org.springframework.web.servlet.view.JstlView"** />  
 <!-- 动态页面的前缀 -->  
 <**property** *name***="prefix"** *value***="/WEB-INF/jsp/"** />  
 <!-- 动态页面的后缀 -->  
 <**property** *name***="suffix"** *value***=".jsp"** />  
</**bean**>

3.4、过滤静态资源

<**mvc:resources** *location***="/css/"** *mapping***="/css/\*\*"** />  
<**mvc:resources** *location***="/js/"** *mapping***="/js/\*\*"** />

整体代码：

<?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:p=*"http://www.springframework.org/schema/p"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*https://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*https://www.springframework.org/schema/context/spring-context.xsd*

*http://www.springframework.org/schema/mvc*

*https://www.springframework.org/schema/mvc/spring-mvc.xsd*

*"*>

<!-- 开启注解扫描 -->

<context:component-scan base-package=*"com.fzq.web"*/>

<!-- 1.映射器: -->

<bean

class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"*></bean>

<!-- 2.适配器: -->

<bean

class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"*></bean>

<!-- 3.视图解析器: -->

<bean

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<!-- 解析jstl标签 -->

<property name=*"viewClass"*

value=*"org.springframework.web.servlet.view.JstlView"* />

<!-- 动态页面的前缀 -->

<property name=*"prefix"* value=*"/WEB-INF/jsp/"* />

<!-- 动态页面的后缀 -->

<property name=*"suffix"* value=*".jsp"* />

</bean>

<!-- 4.过滤静态资源 -->

<mvc:resources location=*"/css/"* mapping=*"/css/\*\*"* />

<mvc:resources location=*"/js/"* mapping=*"/js/\*\*"* />

</beans>

第二步：创建web.xml

3.5.1中央控制器

<display-name>EdocManagement</display-name>

<servlet>

<servlet-name>springmvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<!-- 参数 :告诉中央处理器 使用哪个映射器，哪个适配器，哪个视图解析器 -->

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-web.xml</param-value>

</init-param>

</servlet>

3.5.2引入sping-mvc.xml

<**servlet-mapping**>  
 <**servlet-name**>springmvc</**servlet-name**>  
 <!-- / 中央处理器处理所有请求：包括jsp html 注意：不能写/\*,否则报错 \*.do 中央处理器处理url中以.do结果 比如：  
 http://127.0.0.1:8080/easybuy/index.do 被处理 http://127.0.0.1:8080/easybuy/index.jsp  
 不被处理 -->  
 <!-- <url-pattern>\*.do</url-pattern> -->  
 <**url-pattern**>/</**url-pattern**>  
</**servlet-mapping**>

3.5.3引入context.xml使用监听器加载（放最前面）

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:context.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

3.5.4过滤器设置编码（需要放在控制器上面）

<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>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

整体代码：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://java.sun.com/xml/ns/javaee"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"*

id=*"WebApp\_ID"* version=*"3.0"*>

<!-- 一、spring的配置 -->

<!-- 加载spring容器 -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:context.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 三、spring的编码过滤器 -->

<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>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!--二、 spring mvc -->

<display-name>EdocManagement</display-name>

<servlet>

<servlet-name>springmvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<!-- 参数 :告诉中央处理器 使用哪个映射器，哪个适配器，哪个视图解析器 -->

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-web.xml</param-value>

</init-param>

</servlet>

<!-- 引入spring-web.xml -->

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<!-- / 中央处理器处理所有请求：包括jsp html 注意：不能写/\*,否则报错 \*.do 中央处理器处理url中以.do结果 比如：

http://127.0.0.1:8080/easybuy/index.do 被处理 http://127.0.0.1:8080/easybuy/index.jsp

不被处理 -->

<!-- <url-pattern>\*.do</url-pattern> -->

<url-pattern>/</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

</web-app>