SSM框架整合(雷神278,279集)

Spring+SpringMVC+MyBatis

1. 导包

1.1. Spring

> 【AOP核心】
>
> com.springsource.net.sf.cglib-2.2.0.jar
> com.springsource.org.aopalliance-1.0.0.jar
> com.springsource.org.aspectj.weaver-1.6.8.RELEASE.jar
>
> 【IOC核心】
>
> commons-logging-1.1.3.jar
> spring-aop-4.0.0.RELEASE.jar
> spring-aspects-4.0.0.RELEASE.jar
> spring-beans-4.0.0.RELEASE.jar
> spring-context-4.0.0.RELEASE.jar
> spring-core-4.0.0.RELEASE.jar
> spring-expression-4.0.0.RELEASE.jar
>
> 【JDBC核心】
>
> spring-jdbc-4.0.0.RELEASE.jar
> spring-orm-4.0.0.RELEASE.jar
> spring-tx-4.0.0.RELEASE.jar
>
> 【测试核心】
>
> spring-test-4.0.0.RELEASE.jar

1.2. SpringMVC

> 【ajax】
>
> jackson-annotations-2.1.5.jar
> jackson-core-2.1.5.jar
> jackson-databind-2.1.5.jar
>
> 【数据校验】JSP303校验
>
> hibernate-validator-5.0.0.CR2.jar
> hibernate-validator-annotation-processor-5.0.0.CR2.jar
> classmate-0.8.0.jar
> jboss-logging-3.1.1.GA.jar
> validation-api-1.1.0.CR1.jar
>
> 【上传下载】

> commons-fileupload-1.2.1.jar
> commons-io-2.0.jar
>
> 【jstl-jsp标准标签库】
>
> jstl.jar
> standard.jar
>
> 【验证码】
>
> kaptcha-2.3.2.jar
>
> 【SpringMVC核心】
>
> spring-web-4.0.0.RELEASE.jar
> spring-webmvc-4.0.0.RELEASE.jar

1.3. MyBatis

> 【MyBatis核心】
>
> mybatis-3.4.1.jar
> mybatis-spring-1.3.0.jar【和Spring整合包】
>
> 【ehcache整合】
> ehcache-core-2.6.8.jar
> mybatis-ehcache-1.0.3.jar
> log4j-1.2.17.jar
> slf4j-api-1.7.21.jar
> slf4j-log4j12-1.7.21.jar

1.4. 其他jar包

> mysql-connector-java-5.1.37-bin.jar
> c3p0-0.9.1.2.jar

2. 写配置

2.1. web配置

```xml
<?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">
  <!-- 工具编辑部署描述符，display-name元素包含的就是XML编辑器显示的名称。就是说明的跟你配置不相关 -->
  <display-name>test3_demo</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
   <!-- 配置Spring容器启动:  -->
   <!-- needed for ContextLoaderListener -->
    <context-param>
```

```xml
        <param-name>contextConfigLocation</param-name>
        <!-- 指定Spring配置文件位置 -->
        <param-value>classpath:spring/applicationContext.xml</param-value>
    </context-param>

    <!-- Bootstraps the root web application context before servlet initialization -->
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!-- 配置springMVC前端控制器 -->
    <!-- The front controller of this Spring Web application, responsible for handling all application requests -->
    <servlet>
        <servlet-name>springDispatcherServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spring/applicationContext-mvc.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <!-- Map all requests to the DispatcherServlet for handling -->
    <servlet-mapping>
        <servlet-name>springDispatcherServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <!-- 两个标准配置 -->
    <!-- 字符编码 -->
    <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>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>CharacterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <!-- 支持Rest风格的过滤器 -->
    <filter>
        <filter-name>HiddenHttpMethodFilter</filter-name>
        <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>HiddenHttpMethodFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

</web-app>

```

2.2. Spring配置

```xml
<?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:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd
        http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd">
    <!-- Spring除过控制器不要,剩下的业务逻辑组件都要,包括dao,包括service -->
    <context:component-scan base-package="com.Ray">
        <!-- 扫描排除不写use-default-filters="false" -->
        <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <!-- 0.导入外部配置文件 -->
    <context:property-placeholder location="classpath:dbconfig.properties"/>
    <!-- 1.配数据源 -->
    <bean id="ds" class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="user" value="${jdbc.username}"></property>
        <property name="password" value="${jdbc.password}"></property>
        <property name="driverClass" value="${jdbc.driverclass}"></property>
        <property name="jdbcUrl" value="${jdbc.jdbcurl}"></property>
        <property name="maxPoolSize" value="${jdbc.maxPoolSize}"></property>
        <property name="minPoolSize" value="${jdbc.minPoolSize}"></property>
    </bean>

    <!-- 2.配置JdbcTemplate操作数据库.pass -->
    <!-- 3.配置使用mybatis操作数据库 -->
    <!-- 可以根据配置文件得到sqlSessionFactory -->
    <bean id="" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="configLocation" value="classpath:mybatis:mybatis-config.xml"></property>
        <property name="dataSource" ref="ds"></property>
        <!-- 指定xml映射文件的位置 -->
        <property name="mapperLocations" value="classpath:mybatis/mapper/*.xml"></property>
    </bean>
    <!-- 我们要把每一个dao接口的实现加入到ioc容器 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 指定dao接口所在的包 --><!-- 配置文件用"/",java用"." -->
        <property name="basePackage" value="com.Ray.dao"></property>
    </bean>

    <!-- 4.配置事务控制; 配置事务管理器, 让他控制住数据源里面的链接的关闭和提交-->
    <bean id="tm" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="ds"></property>

    </bean>

    <!-- 5.基于xml配置,配置事务; 哪些方法切入事务还要写切入点表达式 -->
    <aop:config>
    <!-- 配置切入点表达式 -->
        <aop:pointcut expression="execution(* com.Ray.service.*.*(..))" id="txPoint"/>
        <aop:advisor advice-ref="" pointcut-ref="txPoint"/>
    </aop:config>

    <!-- 6.配置事务增强, 事物属性, 事务建议
        transaction-manager="tm", 指定要配置的事务管理器的id
     -->
    <tx:advice id="myTx" transaction-manager="tm">
        <tx:attributes>
            <tx:method name="*" rollback-for="java.lang.Exception"/>
            <tx:method name="get*" read-only="true"/>
            <tx:method name="insertEmp" isolation="READ_UNCOMMITTED"/><!--读取未提交的事务 -->
        </tx:attributes>
    </tx:advice>
</beans>

```

2.3. SpringMVC配置

```xml
<?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:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">
    <!-- SpringMVC只扫描控制器; 禁用默认的规则 -->
    <context:component-scan base-package="com.Ray" use-default-filters="false">
        <context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/pages/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>
    <!-- 配置文件上传解析器 -->
    <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <property name="defaultEncoding" value="utf-8"></property>
        <property name="maxUploadSize" value="#{1024*1024*20}"></property>
        </bean>
    <!-- 扫静态资源 -->
    <mvc:default-servlet-handler/>
    <!-- 扫动态 -->
    <mvc:annotation-driven/>
</beans>
```

2.4. mybatis的配置

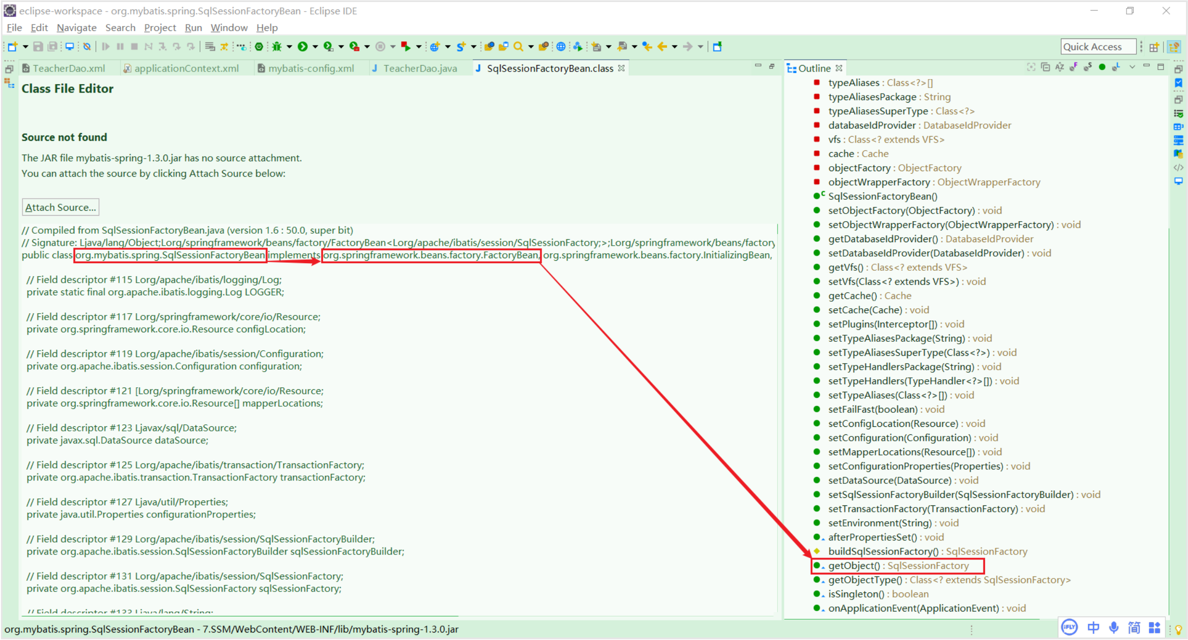
(Spring与mybatis整合包的关键配置)

mybatis的核心就是sqlSessionFactory要根据全局配置文件(mybatis-config),

创建sqlSession对象,把sqlSession拿到映射器进行增删改查

在applicationContext.xml中:

```
<!-- 2.配置JdbcTemplate操作数据库.pass 现在用mybatis-->
    <!-- 3.配置使用mybatis操作数据库 -->
    <!-- 可以根据配置文件得到sqlSessionFactory -->
    <bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="configLocation" value="classpath:mybatis/mybatis-config.xml"></property>
        <property name="dataSource" ref="ds"></property>
        <!-- mybatis-config能写的东西,这里都可以写,完全可以去掉它,只是为了方便,将mybatis-config保留 -->
        <!-- 指定xml映射文件的位置,就不用看mapper批量注册的那个了 -->
        <property name="mapperLocations" value="classpath:mybatis/mapper/*.xml"></property>
    </bean>
    <!-- 我们要把每一个dao接口的实现加入到ioc容器 ,加了就可以AutoWire dao了-->
    <!-- 不这样写的话,可以在TeacherService里面把SqlSessionFactory@AutoWire 就可以,但显然自动注入dao方便操作-->
    <!-- 整合包,可以自动扫描每一个接口的实现 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 指定dao接口所在的包 --><!-- 配置文件用"/",java用"." -->
        <property name="basePackage" value="com.Ray.dao"></property>
    </bean>

```



eclipse-workspace - org.mybatis.spring.SqlSessionFactoryBean - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Quick Access
TeacherDao.xml
applicationContext.xml
mybatis-config.xml
TeacherDao.java
SqlSessionFactoryBean.class
Outline
Attach Source...

Class File Editor

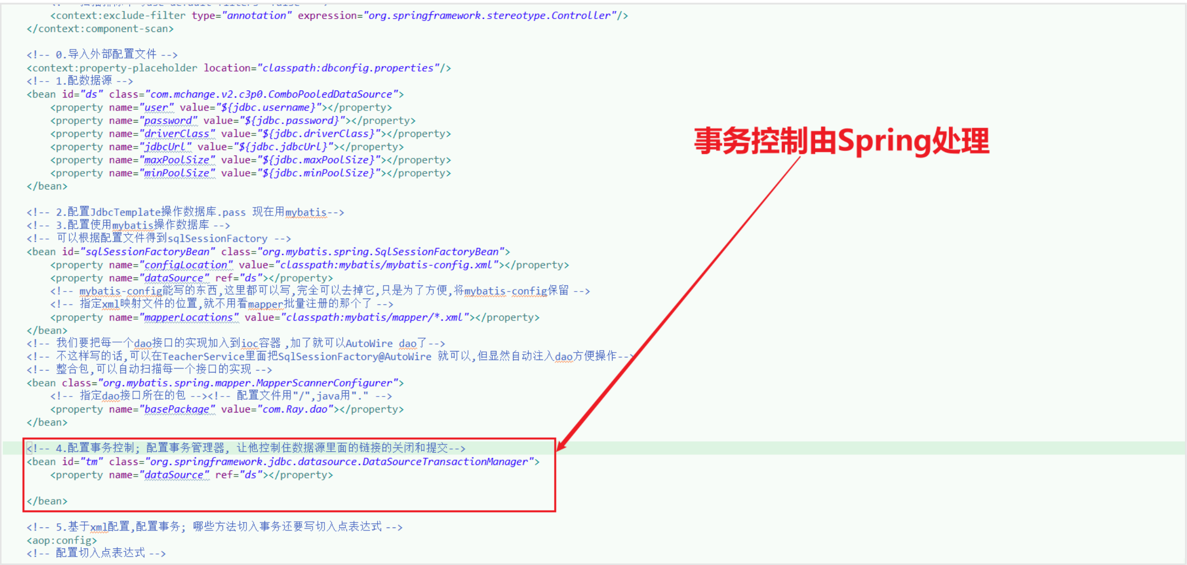
Source not found

The JAR file mybatis-spring-1.3.0.jar has no source attachment.
You can attach the source by clicking Attach Source below:

```
// Compiled from SqlSessionFactoryBean.java (version 1.6 : 50.0, super bit)
// Signature: Ljava/lang/Object;Lorg/springframework/beans/factory/FactoryBean<Lorg/apache/ibatis/session/SqlSessionFactory;>;Lorg/springframework/beans/factory
public class org.mybatis.spring.SqlSessionFactoryBean implements org.springframework.beans.factory.FactoryBean, org.springframework.beans.factory.InitializingBean,

  // Field descriptor #115 Lorg/apache/ibatis/logging/Log;
  private static final org.apache.ibatis.logging.Log LOGGER;

  // Field descriptor #117 Lorg/springframework/core/io/Resource;
  private org.springframework.core.io.Resource configLocation;

  // Field descriptor #119 Lorg/apache/ibatis/session/Configuration;
  private org.apache.ibatis.session.Configuration configuration;

  // Field descriptor #121 [Lorg/springframework/core/io/Resource;
  private org.springframework.core.io.Resource[] mapperLocations;

  // Field descriptor #123 Ljavax/sql/DataSource;
  private javax.sql.DataSource dataSource;

  // Field descriptor #125 Lorg/apache/ibatis/transaction/TransactionFactory;
  private org.apache.ibatis.transaction.TransactionFactory transactionFactory;

  // Field descriptor #127 Ljava/util/Properties;
  private java.util.Properties configurationProperties;

  // Field descriptor #129 Lorg/apache/ibatis/session/SqlSessionFactoryBuilder;
  private org.apache.ibatis.session.SqlSessionFactoryBuilder sqlSessionFactoryBuilder;

  // Field descriptor #131 Lorg/apache/ibatis/session/SqlSessionFactory;
  private org.apache.ibatis.session.SqlSessionFactory sqlSessionFactory;

  // Field descriptor #133 Ljava/lang/String;
```

```
typeAliases : Class<?>[]
typeAliasesPackage : String
typeAliasesSuperType : Class<?>
databaseIdProvider : DatabaseIdProvider
vfs : Class<? extends VFS>
cache : Cache
objectFactory : ObjectFactory
objectWrapperFactory : ObjectWrapperFactory
SqlSessionFactoryBean()
setObjectFactory(ObjectFactory) : void
setObjectWrapperFactory(ObjectWrapperFactory) : void
getDatabaseIdProvider() : DatabaseIdProvider
setDatabaseIdProvider(DatabaseIdProvider) : void
getVfs() : Class<? extends VFS>
setVfs(Class<? extends VFS>) : void
getCache() : Cache
setCache(Cache) : void
setPlugins(Interceptor[]) : void
setTypeAliasesPackage(String) : void
setTypeAliasesSuperType(Class<?>) : void
setTypeHandlersPackage(String) : void
setTypeHandlers(TypeHandler<?>[]) : void
setTypeAliases(Class<?>[]) : void
setFailFast(boolean) : void
setConfigLocation(Resource) : void
setConfiguration(Configuration) : void
setMapperLocations(Resource[]) : void
setConfigurationProperties(Properties) : void
setDataSource(DataSource) : void
setSqlSessionFactoryBuilder(SqlSessionFactoryBuilder) : void
setTransactionFactory(TransactionFactory) : void
setEnvironment(String) : void
afterPropertiesSet() : void
buildSqlSessionFactory() : SqlSessionFactory
getObject() : SqlSessionFactory
getObjectType() : Class<? extends SqlSessionFactory>
isSingleton() : boolean
onApplicationEvent(ApplicationEvent) : void
```

org.mybatis.spring.SqlSessionFactoryBean - 7.SSM/WebContent/WEB-INF/lib/mybatis-spring-1.3.0.jar



事务控制由Spring处理

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

    <!-- 0.导入外部配置文件 -->
    <context:property-placeholder location="classpath:dbconfig.properties"/>
    <!-- 1.配数据源 -->
    <bean id="ds" class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="user" value="${jdbc.username}"></property>
        <property name="password" value="${jdbc.password}"></property>
        <property name="driverClass" value="${jdbc.driverClass}"></property>
        <property name="jdbcUrl" value="${jdbc.jdbcUrl}"></property>
        <property name="maxPoolSize" value="${jdbc.maxPoolSize}"></property>
        <property name="minPoolSize" value="${jdbc.minPoolSize}"></property>
    </bean>

    <!-- 2.配置JdbcTemplate操作数据库.pass 现在用mybatis-->
    <!-- 3.配置使用mybatis操作数据库 -->
    <!-- 可以根据配置文件得到sqlSessionFactory -->
    <bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="configLocation" value="classpath:mybatis/mybatis-config.xml"></property>
        <property name="dataSource" ref="ds"></property>
        <!-- mybatis-config能写的东西,这里都可以写,完全可以去掉它,只是为了方便,将mybatis-config保留 -->
        <!-- 指定xml映射文件的位置,就不用看mapper批量注册的那个了 -->
        <property name="mapperLocations" value="classpath:mybatis/mapper/*.xml"></property>
    </bean>
    <!-- 我们要把每一个dao接口的实现加入到ioc容器 ,加了就可以AutoWire dao了-->
    <!-- 不这样写的话,可以在TeacherService里面把SqlSessionFactory@AutoWire 就可以,但显然自动注入dao方便操作-->
    <!-- 整合包,可以自动扫描每一个接口的实现 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 指定dao接口所在的包 --><!-- 配置文件用"/",java用"." -->
        <property name="basePackage" value="com.Ray.dao"></property>
    </bean>

    <!-- 4.配置事务控制; 配置事务管理器, 让他控制住数据源里面的链接的关闭和提交-->
    <bean id="tm" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="ds"></property>

    </bean>

    <!-- 5.基于xml配置,配置事务; 哪些方法切入事务还要写切入点表达式 -->
    <aop:config>
    <!-- 配置切入点表达式 -->
```

2.5. 整合步骤:

2.5.1. 导入整合包;(能将dao的实现加入到容器中)

3. 测试

```
package com.Ray.dao;

import com.Ray.bean.Teacher;

public interface TeacherDao {
    public Teacher getTeacherById(Integer i);
}
```