SSM

# 概述

SSM（Spring + SpringMVC + MyBatis）的整合，其中使用MyBatis的方式一般有两种。

* DAO与DAO实现类的方式
* Mapper接口方式

# DAO与DAO实现类的方式

此种方式的主要不同点是DAO实现类的写法、applicationContext.xml的配置不一样。

1. DAO实现类要继承SqlSessionDaoSupport，目的是简单化。（DAO实现类中不用写sqlSessionFactory字段来接收注入，因为父类有这个字段，往父类中注入即可；不用手动获取对应的mapper，因为父类中已经做了这个工作。）

public class UserDaoImpl extends SqlSessionDaoSupport implements UserDao {  
 public UserDO getUser(Integer id) {  
 SqlSession sqlSession = this.getSqlSession();//从工厂处获取一个session  
 UserDO user = sqlSession.selectOne("test.getUser",id);//执行查询  
 return user;  
 }  
}

这里的sqlSession不用手动关闭，否则报错。

1. applicationContext.xml中要配置DAO实现类，且以属性的形式注入sqlSessionFactory。

<bean class="com.demo.dao.user.impl.UserDaoImpl">  
 <property name="sqlSessionFactory" ref="sessionFactory"/>  
</bean>

# Mapper接口方式

Mapper接口方式的主要不同点是不用写DAO实现类了，mybatis.xml、applicationApplication.xml的配置不同DAO方式：

1. 不用写DAO实现类
2. mybatis.xml不用写<mapper.../>元素

<?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">  
<configuration>  
 <!-- 全局参数 -->  
 <settings>

//.....  
 </settings>  
</configuration>

1. applicationContext.xml不用写具体的DAO实现类，指定Mapper.xml和其它对应接口即可，具体对应SqlSessionFactoryBean、mapperScannerConfigurer。

<context:property-placeholder location="classpath:config.properties"/>  
<context:component-scan base-package="com.demo">  
 <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>  
</context:component-scan>  
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"  
 p:driverClass="${jdbc.mysql.driverClass}"  
 p:jdbcUrl="${jdbc.mysql.url}"  
 p:user="${jdbc.mysql.user}"  
 p:password="${jdbc.mysql.password}"  
 p:maxIdleTime="${jdbc.maxIdleTime}"  
 p:maxPoolSize="${jdbc.minPoolSize}"  
 p:minPoolSize="${jdbc.maxPoolSize}"  
 p:initialPoolSize="${jdbc.initialPoolSize}">  
</bean>  
<bean id="sessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"  
 p:dataSource-ref="dataSource"  
 p:configLocation="classpath:config/mybatis.xml"  
 p:mapperLocations="classpath:mappers/user/UserDaoMapper.xml">  
</bean>  
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer" id="mapperScannerConfigurer"  
 p:sqlSessionFactoryBeanName="sessionFactory"  
 p:basePackage="com.demo.dao.user">  
</bean>

Mapper接口方式需要遵循如下规则：

1. Mapper接口类全名和Mapper.xml文件的namespace名称相同。

在applicationContext.xml文件中已经配置了Mapper接口类全名和Mapper.xml文件，并交给了对应的处理器。

# SqlSessionTemplate方式

这种方式配置起来更加简单。

不需要DAO接口，只需要在applicationContext.xml中配置扫描的Mapper.xml和SqlSessionTemplate即可。

1、applicationContext.xml配置：

<context:property-placeholder location="classpath:config.properties"/>  
<context:component-scan base-package="com.demo">  
 <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>  
</context:component-scan>  
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"  
 p:driverClass="${jdbc.mysql.driverClass}"  
 p:jdbcUrl="${jdbc.mysql.url}"  
 p:user="${jdbc.mysql.user}"  
 p:password="${jdbc.mysql.password}"  
 p:maxIdleTime="${jdbc.maxIdleTime}"  
 p:maxPoolSize="${jdbc.minPoolSize}"  
 p:minPoolSize="${jdbc.maxPoolSize}"  
 p:initialPoolSize="${jdbc.initialPoolSize}">  
</bean>  
<bean id="sessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"  
 p:dataSource-ref="dataSource"  
 p:configLocation="classpath:config/mybatis.xml"  
 p:mapperLocations="classpath\*:mappers/\*\*/\*.xml">  
</bean>  
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">  
 <constructor-arg index="0" ref="sessionFactory" />  
</bean>

1. Mapper.xml配置

<mapper namespace="com.demo.service.user.impl.UserServiceImpl">  
 <select id="getUser" resultType="com.demo.model.DO.UserDO">  
 SELECT t.id,t.address,t.birthday,t.sex,t.username FROM user t WHERE t.id = #{id}  
 </select>  
</mapper>

namespace名称跟Service类全名一致。

3、service类：

private static String NAMESPACE = "com.demo.service.user.impl.UserServiceImpl.";

@Resource

private SqlSessionTemplate sqlSessionTemplate;

public UserDO getUser(Integer id) {

return sqlSessionTemplate.selectOne(NAMESPACE + "getUser",id);

}

使用sqlSessionTemplate执行sql操作，注意是 namespace+Mapper.xml文件中的方法名。