SSM 配置文件 分析

spring 配置文件（主要整合的是 spring 和 mybatis 的配置文件）

问题： 两者之间没有整合在一起的时候是怎么样的

spring 配置文件： Spring 配置文件是用于指导 Spring 工厂进行 Bean 生产、依赖关系注入（装配）及 Bean 实例分发的"图纸"。Java EE 程序员必须学会并灵活应用这份"图纸"准确地表达自己的"生产意图"。Spring 配置文件是一个或多个标准的 XML 文档，applicationContext.xml 是 Spring 的默认配置文件，当容器启动时找不到指定的配置文档时，将会尝试加载这个默认的配置文件。

```
<?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"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="duke" class="com.springinaction.springidol.Juggler" >
        <!-- 通过构造方法设置属性值 -->
        <constructor-arg value="15"></constructor-arg>
    </bean>

    <bean id="sonnect29" class="com.springinaction.springidol.Sonnet29"></bean>

    <bean id="poeticPoem" class="com.springinaction.springidol.PoeticJuggler">
        <constructor-arg value="15"></constructor-arg>
        <constructor-arg ref="sonnect29"></constructor-arg>
    </bean>

    <bean id="saxphone" class="com.springinaction.springidol.saxphone"></bean>
    <bean id="piano" class="com.springinaction.springidol.piano"></bean>

    <!-- p 命名空间用法 -->
    <bean id="Kenny2" class="com.springinaction.springidol.Instrumentalist"
        p:song="Lemon Tree" p:age="30" p:instrument-ref="saxphone"     >
    </bean>

    <!-- 为集合配置 bean -->
    <bean id="hank" class="com.springinaction.springidol.OneManBand">
        <property name="instruments">
            <list>
                <ref bean="piano" />
                <ref bean="saxphone" />
            </list>
```

```
        </property>
        <property name="instruments2">
            <map>
                <entry key="piano" value-ref="piano"></entry>
                <entry key="saxphone" value-ref="saxphone"></entry>
            </map>
        </property>
    </bean>

    <!-- properties 的写法 -->
    <bean id="hank2" class="com.springinaction.springidol.OneManBand">
        <property name="instruments">
            <props>
                <!-- key 和 value 都为 String -->
                <prop key="piano">la la la</prop>
                <prop key="saxphone">ta ta ta</prop>
            </props>
        </property>
    </bean>

    <!-- 赋 null 值 -->
    <!--
    ...
        <property name="xxx"><null/></property>
    ...
     -->
</beans>
```

个人理解：在最基础的 spring 配置文件就是做控制反转的功能，就是将实体类的创建放在 spring beanfactory 中
spring 里面配置，就是围绕着各种 bean，有 bean 的 id， bean 对应的 class，以及实现这个 class 里面
的一些注入 property， 就像上面红色的所示。 把一个个需要 beanfactory 工厂创建的 bean 告诉 beanfactory
工厂。

mybatis 配置文件：

```
<?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>
```

```
    <!-- 加载类路径下的属性文件 -->
    <properties resource="db.properties"/>

    <!-- 设置一个默认的连接环境信息 -->
    <environments default="mysql_developer">
        <environment id="mysql_developer">
            <!-- mybatis 使用 jdbc 事务管理方式 -->
            <transactionManager type="JDBC"></transactionManager>
            <!-- mybatis 使用连接池方式来获取连接 -->
            <dataSource type="POOLED">
                <!-- 配置与数据库交互的 4 个必要属性，不要直接写，单独写在一个配置文件中 -->
                <property name="driver" value="${sqlserver.driver}"/>
                <property name="url" value="${mysql.url}"/>
                <property name="username" value="${mysql.username}"/>
                <property name="password" value="${mysql.password}"/>
            </dataSource>
        </environment>
        <!-- 连接环境信息，取一个任意唯一的名字 -->
    </environments>

    <!-- 加载映射文件-->
    <mappers>
        <mapper resource="com/winner/entity/StudentMapper.xml"/>
    </mappers>
</configuration>
```

对应的 sql xml 文件：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<!--namespace 可以写类的全限定名，这样做的好处是
    sqlSession.insert(Student.class.getName()+".addStudent");
-->
<mapper namespace="com.winner.entity.StudentMapper">

    <!--实体与表的映射，type 是类名，但是没有表名，可以理解表名在下面的 sql 语句中-->
    <resultMap id="studentMap" type="com.winner.entity.Student">
        <id property="id" column="id"/>
        <result property="name" column="name"/>
        <result property="sal" column="sal"/>
    </resultMap>
    <insert id="addStudent" parameterType="com.winner.entity.Student">
```

```
        <![CDATA[
            INSERT INTO student(id,name,sal) VALUES (#{id},#{name},#{sal})
        ]]>
    </insert>
</mapper>
```

个人理解： mybatis 配置文件，一个是配置环境（连接数据库的配置， 以及 mapper 文件的配置）， 一个是用来编写
sql 语句的 xml 文件（用于实体类中与其相互匹配）， 然后编写对应的 dao 类来执行操作。这个部分介绍
有待深入

spring 和 mybatis 整合配置：

```
<?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:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">
    <!--读取 db.properties -->
    <!-- source 文件夹下面存放的就是 classpath 文件， 系统可以找到的 -->
    <!-- 这个文件里面主要就是配置 bean 工厂，因为 spring 就是一个 bean 工厂，然后将数据库
mybatis 整合在里面 -->
    <!-- 一个是获取 jdbc 需要的一些东西， 然后就是配置 mybatis 里面需要的东西-->

    <context:property-placeholder location="classpath:db.properties"/>
    <!-- 配置数据源 -->
    <bean id="dataSource"
        class="org.apache.commons.dbcp2.BasicDataSource">
        <!--数据库驱动 -->
        <property name="driverClassName" value="${jdbc.driver}" />
        <!--连接数据库的 url -->
```

```
    <property name="url" value="${jdbc.url}" />
    <!--连接数据库的用户名 -->
    <property name="username" value="${jdbc.username}" />
    <!--连接数据库的密码 -->
    <property name="password" value="${jdbc.password}" />
    <!--最大连接数 -->
    <property name="maxTotal" value="${jdbc.maxTotal}" />
    <!--最大空闲连接 -->
    <property name="maxIdle" value="${jdbc.maxIdle}" />
    <!--初始化连接数 -->
    <property name="initialSize" value="${jdbc.initialSize}" />
</bean>


<!-- 事务管理器 -->
<bean id="transactionManager" class=
"org.springframework.jdbc.datasource.DataSourceTransactionManager">
<!-- 数据源 -->
<property name="dataSource" ref="dataSource" />
</bean>
    <!-- 通知 -->
    <tx:advice id="txAdvice" transaction-manager="transactionManager">
    <tx:attributes>
    <!-- 传播行为 -->
    <tx:method name="save*" propagation="REQUIRED" />
    <tx:method name="insert*" propagation="REQUIRED" />
    <tx:method name="add*" propagation="REQUIRED" />
    <tx:method name="create*" propagation="REQUIRED" />
    <tx:method name="delete*" propagation="REQUIRED" />
    <tx:method name="update*" propagation="REQUIRED" />
    <tx:method name="find*" propagation="SUPPORTS" read-only="true" />
    <tx:method name="select*" propagation="SUPPORTS" read-only="true" />
    <tx:method name="get*" propagation="SUPPORTS" read-only="true" />
    </tx:attributes>
    </tx:advice>
    <!-- 切面 -->
    <aop:config>
<aop:advisor advice-ref="txAdvice"
pointcut="execution(* com.qf.core.service.*.*(..))" />
</aop:config>


<!-- 配置 MyBatis 的工厂 -->
<bean class="org.mybatis.spring.SqlSessionFactoryBean">
```

```
        <!-- 数据源 -->
        <property name="dataSource" ref="dataSource" />
        <!-- 配置 MyBatis 的核心配置文件所在位置 -->
        <property name="configLocation" value="classpath:mybatis-config.xml" />
    </bean>

    <  !-- the below configuration told the things that the sql sentence and java function -->
    <!-- the function for below configuration is down for mybatis -->
    <!-- 配置 mapper 扫描器 接口开发,扫描 com.qf.core.dao 包 ， 写在此包下的接口即可被扫描到 --
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.qf.core.dao" />
    </bean>


    <!-- 配置扫描@Service 注解 -->
        <context:component-scan base-package="com.qf.core.service"/>
  </beans>
```

个人理解： 整合之后将原来在 mybatis 配置文件搭建环境的部分，连接数据库，mapper 部分（整合之后的 mapper

不需要像之前写每一个 mappper， 而是直接扫描存放 mapper 文件（sql.xml 文件）的包就可以了，包里面 存 放对应的 dao 就可以了）都在 bean 中完成了。原来需要一个个放入 bean factory 的 bean 现在可以直接通过

扫描一个包就可以了， 就像上面配置扫描@service 注解 一样

整合后的 mybatis 配置文件：

```
<?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>
<!-- 原来是将数据库连接放在这里，然后就是对应的 po 实体类对应建立映射 -->
<!-- 将实体类和数据库里面的表格之间构建连接 -->
<!-- 别名定义 -->
<typeAliases>
<package name="com.qf.core.po" />
</typeAliases>
</configuration>
```

个人理解：活基本上让 spring 干完了，现在 mybatis 配置文件里面做的就 比较少了