mybatis和spring

作者：*杨旭东*
时间：*8/15/2017 11:29:41 AM *

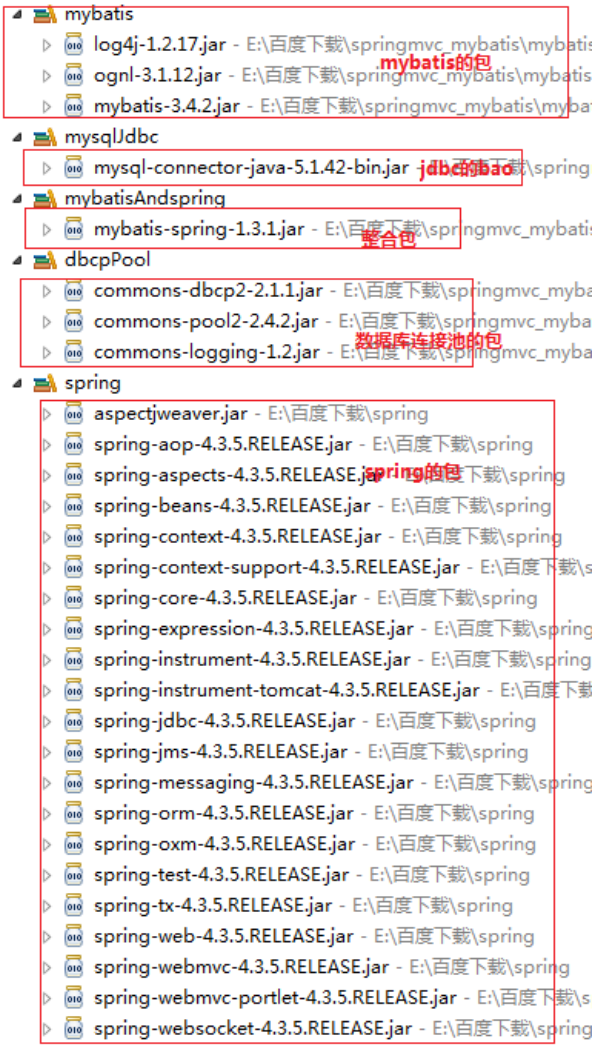
场景

spring作为容器框架，通常在web开发的时候，我们会将持久层框架（mybatis，hibernate..），应用层框架(struts(1x,2x),springMvc...)，以及我们的业务层与容器框架(spring)进行整合。形成ssh或ssm架构方便管理，同时增强代码的扩展性和mvc的结构化。

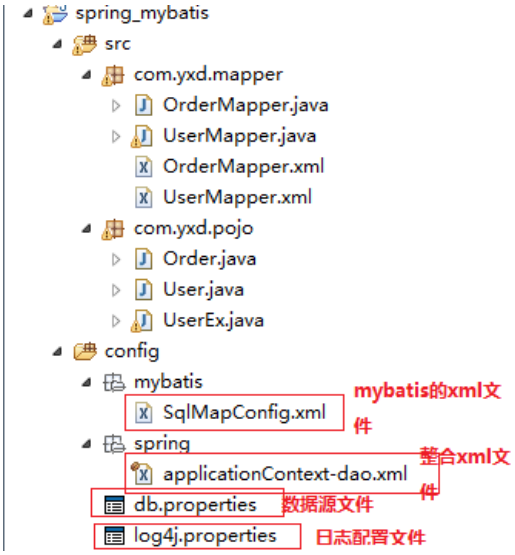
补充：建议学完spring和mybatis再看本部分，当然没有过spring也能看懂，为了以后的思想层面增强，建议看完两者看完再去看整合。

所需jar包

- mybatis的jar
- JDBC的jar（案例用的mysql）
- spring和mybatis的整合包
- spring的jar
- 数据库连接池的jar（数据库连接池有很多。例如c3p0，dbcp[案例使用]，jdbc默认的....）
- commings-logging的jar
- log4j的jar
- junit4的jar包（测试）



文件配置



sqlMpConfig.xml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>
    <settings>
        <!-- 开启懒加载 -->
        <setting name="lazyLoadingEnabled" value="true" />
        <setting name="aggressiveLazyLoading" value="false" />
    </settings>
    <typeAliases>
        <!-- 批量扫描用别名 -->
        <package name="com.yxd.pojo" />
    </typeAliases>
    <mappers>
        <!--扫描mapper 此种方法要求mapper接口名称和mapper映射文件名称相同，且放在同一个目录中。 -->
        <package name="com.yxd.mapper"/>
    </mappers>
</configuration>
```

applicationContext-dao.xml文件

```
<?xml version="1.0" encoding="UTF-8"?>
<!--spring遵循的一些规范 -->
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx" xmlns:p="http://www.springframework.org/schema/p"
    xmlns:util="http://www.springframework.org/schema/util" xmlns:jdbc="http://www.springframework.org/schema/jdbc"
    xmlns:cache="http://www.springframework.org/schema/cache"
    xmlns:context.="http://www.springframework.org/schema/util"
    xsi:schemaLocation="
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd
http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.1.xsd
http://www.springframework.org/schema/cache
http://www.springframework.org/schema/cache/spring-cache-3.1.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd
```

```
http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util.xsd">
    <!-- 加载db.properties中的内容，db.properties的key命名要有一定的特殊规则。 -->
    <context:property-placeholder location="classpath:db.properties" />

    <!-- 配置数据源，采用的是dbcp的数据库连接池(其他还有c3p0,如果不配置则采用默认的jdbc的数据库连接池) -->
    <bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource"
        destroy-method="close">
        <property name="driverClassName" value="${mysql.driver}" />
        <property name="url" value="${mysql.url}" />
        <property name="username" value="${mysql.username}" />
        <property name="password" value="${mysql.password}" />
        <property name="maxTotal" value="30" />
        <property name="maxIdle" value="5" />
    </bean>

    <!-- 第一步配置mybatis的sqlSessionFactory -->

    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 记载mybatis全局配置文件 -->
        <property name="configLocation" value="classpath:mybatis/SqlMapConfig.xml" />
        <!-- 配置数据源 -->
        <property name="dataSource" ref="dataSource" />
    </bean>
    <!-- 单一mapper的配置 -->
    <!-- <bean id="userMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">
        <property name="mapperInterface" value="com.yxd.mapper.UserMapper"/> <property
        name="sqlSessionFactory" value="sqlSessionFactory"/> </bean> -->

        <!-- 扫描出来的bean的id，是mapper名的小写！！ -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.yxd.mapper" />
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />
    </bean>
</beans>
```

db.properties文件

补充：配置如果显示乱码，将文件字符集设置成utf-8.如果依旧是乱码，自行下载propertis插件，用propertisEditor方式打开。

```
mysql.driver=com.mysql.jdbc.Driver
mysql.url=jdbc:mysql://localhost:3306/demo?emp;useUnicode=true&emp;characterEncoding=utf-8&emp;useSSL=false
mysql.username=root
mysql.password=root
```

log4j.propertis文件

```
# 注意：设置成debug级别的，发布后设置成error,stdout为未来日志的目的地名称
log4j.rootLogger=DEBUG, stdout
lo
# Console控制台设置
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
# %l:添加该属性，可以在日志中看出该条日志在哪里发生的 （一般不用）
log4j.appender.stdout.layout.ConversionPattern= %d{yyyy MMM dd HH:mm:ss}  %p [%t] - %m%n
```

测试代码：

```
package com.yxd.mapper;

import java.io.InputStream;
import java.util.Iterator;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
```

```
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Before;
import org.junit.Test;

import com.yxd.pojo.Order;

/**
 * @Title: OrderMapperTest.java
 * @Package com.yxd.mapper
 * @Description:
 * @author 杨旭东
 * @date 2017年8月10日 下午11:22:46
 * @version V1.0
 */
public class OrderMapperTest {
    private SqlSessionFactory sqlSessionFactory;

    @Before
    public void setUp() throws Exception {
        // mybatis配置文件
        String resource = "mybatis/SqlMapConfig.xml";
        InputStream inputStream = Resources.getResourceAsStream(resource);
        // 使用SqlSessionFactoryBuilder创建sessionFactory
        sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
    }

    @Test
    public void testSlectUserFindOrders() throws Exception {
        SqlSession sqlSession= sqlSessionFactory.openSession();
         OrderMapper orderMapper= sqlSession.getMapper(OrderMapper.class);
         List<Order> order = orderMapper.slectUserFindOrders();
         System.out.println(order.get(1).getUserEx().getUsername());
            Iterator<Order> iterator=  order.iterator();
            while(iterator.hasNext()) {
                System.out.println(iterator.next().getOrderPrice());
            }
         sqlSession.close();
    }
    }
```

总结

——>End(完)