2、Spring整合mybatis

一、主要知识点

在基础的 MyBatis 用法中，是通过 SqlSessionFactoryBuilder 来创建 SqlSessionFactory 的。 而在 MyBatis-Spring 中，则使用 SqlSessionFactoryBean 来创建。

1、sqlSessionFactory

里面有Mybatis里的所有属性，可以完全取代Mybatis配置文件

SqlSessionFactory 有一个唯一的必要属性：用于 JDBC 的 DataSource。这可以是任意的 DataSource 对象，它的配置方法和其它 Spring 数据库连接是一样的。

1）configLocation

它用来指定 MyBatis 的 XML 配置文件路径。它在需要修改 MyBatis 的基础配置非常有用。通常，基础配置指的是 <settings> 或 <typeAliases> 元素。

需要注意的是，这个配置文件并不需要是一个完整的 MyBatis 配置。确切地说，任何环境配置（<environments>），数据源（<DataSource>）和 MyBatis 的事务管理器（<transactionManager>）都会被忽略。

SqlSessionFactoryBean 会创建它自有的 MyBatis 环境配置（Environment），并按要求设置自定义环境的值。

2）mapperLocations

如果 MyBatis 在映射器类对应的路径下找不到与之相对应的映射器 XML 文件，那么也需要配置文件。这时有两种解决办法：

第一种是手动在 MyBatis 的 XML 配置文件中的 <mappers> 部分中指定 XML 文件的类路径；

第二种是设置工厂 bean 的 mapperLocations 属性。

mapperLocations 属性接受多个资源位置。这个属性可以用来指定 MyBatis 的映射器 XML 配置文件的位置。属性的值是一个 Ant 风格的字符串，可以指定加载一个目录中的所有文件，或者从一个目录开始递归搜索所有目录。比如:

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <!--绑定（替换）
    Mybatis配置文件-->
    <property name="configLocation" value="classpath:mybatis-config.xml"/>
    <!--绑定（替换）Mybatis映射文件-->
    <property name="mapperLocations"
```

```
value="classpath:com/kuang/mapper/*.xml"/>
    </bean>
```

二、步骤

1、创建Spring-Dao.xml开始整合

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!--DataSource：使用Spring的数据源替换Mybatis的配置 c3p0 dhcp druid
    我们这里使用Spring提供的JDBC：
org.springframework.jdbc.datasource.DriverManagerDataSource
    -->
    <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://127.0.0.1:3306/mybatis01?
useUnicode=true&characterEncoding=utf8&serverTimezone=GMT"/>
        <property name="username" value="root"/>
        <property name="password" value="qrj15521026074"/>
    </bean>

    <!--替换sqlSessionFactory-->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <!--绑定（替换）Mybatis配置文件-->
        <property name="configLocation" value="classpath:mybatis-config.xml"/>
        <!--绑定（替换）Mybatis映射文件-->
        <property name="mapperLocations"
value="classpath:com/qiu/mapper/*.xml"/>
    </bean>

    <!--SqlSessionTemplate：就是我们使用的sqlSession-->
    <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
        <!--只能使用构造器注入sqlSessionFactory:因为没有set方法-->
        <constructor-arg index="0" ref="sqlSessionFactory"/>
    </bean>

</beans>
```

替换mybatis-config.xml

```
//1.读取配置文件，生成字节输入流
InputStream in = Resources.getResourceAsStream("mybatis-config.xml");
//2.获取SqlSessionFactory
SqlSessionFactory factory = new SqlSessionFactoryBuilder().build(in);
//3.获取SqlSession对象
SqlSession  sqlSession = factory.openSession(true);
```

2、创建UserMapperImpl 实现类

```
public class UserMapperImpl implements UserMapper{

   //我们的所有操作，都使用sqlSession来执行，在原来，现在都使用
SqlSessionTemplate;
   private SqlSessionTemplate sqlSession;

   public void setSqlSession(SqlSessionTemplate sqlSession) {
      this.sqlSession = sqlSession;
   }

   @Override
   public List<User> selectUser() {
      UserMapper mapper = sqlSession.getMapper(UserMapper.class);
      return mapper.selectUser();
   }

}
```

3、将UserMapperImpl注入Spring-Dao.xml中

```
<bean id="UserMapper" class="com.qiu.mapper.UserMapperImpl">
   <property name="sqlSession" ref="sqlSession"/>
</bean>
```

4、测试：此时没有了mybatis的内容

```
public class MyText {
   public static void main(String[] args) {

      ApplicationContext context = new ClassPathXmlApplicationContext("spring-
dao.xml");
      UserMapper userMapper = context.getBean("UserMapper", UserMapper.class);

      for(User user:userMapper.selectUser()){
         System.out.println(user);
      }
   }
}
```

三、改进

1、改进Spring-Dao.xml

可以把userMapper的配置弄到新的一个配置文件

applicationContext.xml里

这样Spring-Dao.xml里的文件就基本是固定的mybatis配置了

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <import resource="spring-dao.xml"/>

    <!---->
    <bean id="userMapper" class="com.kuang.mapper.UserMapperImpl">
        <property name="sqlSession" ref="sqlSession"/>
    </bean>

</beans>
```

2、改进UserMapperImpl

通过继承SqlSessionDaoSupport，使用getSqlSession()方法即可获得一个sqlSession

```
public class UserMapperImpl2  extends SqlSessionDaoSupport implements
UserMapper {
    @Override
    public List<User> selectUser() {
//        SqlSession sqlSession=getSqlSession();
//        UserMapper userMapper=sqlSession.getMapper(UserMapper.class);
//        return userMapper.selectUser();
        return getSqlSession().getMapper(UserMapper.class).selectUser();
    }
}
```

测试时；

```
public class MyText {
    public static void main(String[] args) {

        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        UserMapper userMapper = context.getBean("UserMapper", UserMapper.class);

        for(User user:userMapper.selectUser()){
            System.out.println(user);
        }
    }
}
```