

整合MyBatis

步骤:

- 导入相关jar包 (添加依赖)
 - junit
 - mybatis
 - mysql
 - spring相关的
 - aop织入
 - mybatis-spring 【专门整合spring和mybatis的】

```
1 <dependencies>
2   <dependency>
3     <groupId>junit</groupId>
4     <artifactId>junit</artifactId>
5     <version>4.12</version>
6   </dependency>
7
8   <!--Mysql 驱动-->
9   <dependency>
10     <groupId>mysql</groupId>
11     <artifactId>mysql-connector-java</artifactId>
12     <version>5.1.46</version>
13   </dependency>
14
15   <dependency>
16     <groupId>org.mybatis</groupId>
17     <artifactId>mybatis</artifactId>
18     <version>3.4.6</version>
19   </dependency>
20
21   <dependency>
22     <groupId>org.springframework</groupId>
23     <artifactId>spring-webmvc</artifactId>
24     <version>5.2.0.RELEASE</version>
25   </dependency>
26
27   <!--Spring操作数据库的话，还需要一个Spring-jdbc
28   这里对应 后续配置DataSource
29   -->
30   <dependency>
31     <groupId>org.springframework</groupId>
32     <artifactId>spring-jdbc</artifactId>
33     <version>5.2.0.RELEASE</version>
34   </dependency>
35
36   <dependency>
37     <groupId>org.aspectj</groupId>
38     <artifactId>aspectjweaver</artifactId>
39     <version>1.8.13</version>
40   </dependency>
41
```

```

42     <dependency>
43         <groupId>org.mybatis</groupId>
44         <artifactId>mybatis-spring</artifactId>
45         <version>2.0.2</version>
46     </dependency>
47
48     <dependency>
49         <groupId>org.projectlombok</groupId>
50         <artifactId>lombok</artifactId>
51         <version>1.18.8</version>
52     </dependency>
53 </dependencies>

```

- 编写配置文件
- 测试运行

1 回顾MyBatis, 测试环境

- 实体类

```

1  @Data
2  @AllArgsConstructor
3  @NoArgsConstructor
4  public class User implements Serializable {
5      private int id;
6      private String name;
7      private String pwd;
8  }

```

这里使用了Lombok, 所以没有get、set、toString方法。

- mapper接口

```

1  public interface UserMapper {
2      /**
3       * 查询所有的用户
4       *
5       * @return
6       */
7      public List<User> getUsers();
8  }

```

- mapper.xml

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE mapper
3      PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6  <mapper namespace="com.kuang.mapper.UserMapper">
7      <select id="getUsers" resultType="user">
8          select * from mybatis.user
9      </select>
10
11 </mapper>

```

- 核心配置文件

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE configuration
3     PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-config.dtd">
5
6 <configuration>
7     <typeAliases>
8         <package name="com.kuang.pojo"/>
9     </typeAliases>
10
11     <environments default="development">
12         <environment id="development">
13             <transactionManager type="JDBC"/>
14             <dataSource type="POOLED">
15                 <property name="driver" value="com.mysql.jdbc.Driver"/>
16                 <property name="url"
17                     value="jdbc:mysql://localhost:3306/mybatis?
18 useSSL=true&useUnicode=true&characterEncoding=UTF-8"/>
19                 <property name="username" value="root"/>
20                 <property name="password" value="mynewroot"/>
21             </dataSource>
22         </environment>
23     </environments>
24
25     <mappers>
26         <mapper class="com.kuang.mapper.UserMapper"/>
27     </mappers>
28 </configuration>
```

- 工具类

```
1 public class MybatisUtils {
2     private static SqlSessionFactory sqlSessionFactory;
3
4     static {
5         //使用Mybatis的第一步：获取sqlSessionFactory对象
6         try {
7             String resource = "mybatis-config.xml";
8             InputStream inputStream =
9 Resources.getResourceAsStream(resource);
10             sqlSessionFactory = new
11 SqlSessionFactoryBuilder().build(inputStream);
12         } catch (IOException e) {
13             e.printStackTrace();
14         }
15     }
16
17     // 使用SqlSessionFactory获取SqlSession
18     public static SqlSession getSqlSession() {
19         return sqlSessionFactory.openSession();
20     }
21 }
```

- 测试运行

```
1  @Test
2  public void getUsers(){
3      SqlSession sqlSession = MybatisUtils.getSqlSession();
4
5      UserMapper mapper = sqlSession.getMapper(UserMapper.class);
6      List<User> users = mapper.getUsers();
7
8      for (User user : users) {
9          System.out.println(user);
10     }
11 }
```

```
User(id=1, name=test, pwd=123)
User(id=3, name=jerry, pwd=456789)
User(id=4, name=李三, pwd=123)
User(id=6, name=李明, pwd=ff3rwf)

Process finished with exit code 0
```

2 mybatis—spring (方式一)

官网: <http://mybatis.org/spring/zh/index.html>

介绍:

1. 版本注意

MyBatis-Spring	MyBatis	Spring 框架	Spring Batch	Java
2.0	3.5+	5.0+	4.0+	Java 8+
1.3	3.4+	3.2.2+	2.1+	Java 6+

2. MyBatis-Spring 会帮助你将 MyBatis 代码无缝地整合到 Spring 中。它将允许 MyBatis 参与到 Spring 的事务管理之中, 创建映射器 mapper 和 `SqlSession` 并注入到 bean 中。
3. 所指定的映射器类必须是一个**接口**, 而不是具体的实现类。

2.1 编写核心配置文件

- 这里我们说明一点: 我们会写一个spring-dao.xml, 里面专注于一些固定的配置(数据库、整合mybatis等等)
- 而applicationContext.xml, 是总的配置文件, 我们在这个文件中只做两个任务:
 - 引入spring-dao.xml (在这个配置文件中, 整合了Mybatis的配置, 所以其实我们也是可以省略myabtis的核心配置文件的)
 - 注册管理bean
- applicationContext.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xmlns:aop="http://www.springframework.org/schema/aop"
5         xsi:schemaLocation="http://www.springframework.org/schema/beans
6                             https://www.springframework.org/schema/beans/spring-beans.xsd
```

```

7      http://www.springframework.org/schema/aop
8      https://www.springframework.org/schema/aop/spring-aop.xsd">
9
10     <import resource="spring-dao.xml"/>
11
12     <bean id="userMapper" class="com.kuang.mapperImpl.UserMapperImpl">
13         <!--set方法注入sqlSessionTemplate-->
14         <property name="sqlSessionTemplate" ref="sqlSessionTemplate"/>
15     </bean>
16 </beans>

```

- spring-dao.xml

一共分为三部分：

- DataSource（数据源）
- SqlSessionFactory（里面配置数据源、还有mybatis的配置）
- SqlSessionTemplate（也就是SqlSession），以后不再使用工具类获得SqlSession

重点在SqlSessionFactory和SqlSessionTemplate这两个对象。

DataSource要注入到SqlSessionFactory中。

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:aop="http://www.springframework.org/schema/aop"
5      xsi:schemaLocation="http://www.springframework.org/schema/beans
6          https://www.springframework.org/schema/beans/spring-beans.xsd
7          http://www.springframework.org/schema/aop
8          https://www.springframework.org/schema/aop/spring-aop.xsd">
9
10     <!--
11     DataSource:使用Spring的数据源替换MyBatis的配置
12     c3p0、druid、dbcp
13     我们这里使用Spring提供的JDBC：依赖引入了spring-jdbc
14     -->
15     <bean id="dataSource"
16         class="org.springframework.jdbc.datasource.DriverManagerDataSource">
17         <property name="driverClassName"
18             value="com.mysql.jdbc.Driver"/>
19         <property name="url"
20             value="jdbc:mysql://localhost:3306/mybatis?
21             useSSL=true&useUnicode=true&characterEncoding=UTF-8"/>
22         <property name="username" value="root"/>
23         <property name="password" value="mynewroot"/>
24     </bean>
25
26     <!--sqlSessionFactory
27     记得配置数据源
28     -->
29     <bean id="sqlSessionFactory"
30         class="org.mybatis.spring.SqlSessionFactoryBean">
31         <!--配置数据源-->
32         <property name="dataSource" ref="dataSource"/>
33
34         <!--可以绑定mybatis的配置文件
35         这样子就可以引用mybatis中的一些配置（别名。。。）
36         -->

```

```

33     <property name="configLocation" value="classpath:mybatis-
config.xml"/>
34
35     <!--当然也可以在这里配置myabtis里面的配置
36     例如: mapper
37     一般我们的myabtis的配置文件只留两个: 别名管理 和 setting设置
38     -->
39     <property name="mapperLocations"
value="classpath:com/kuang/mapper/*.xml"/>
40
41     </bean>
42
43     <!--sqlSession
44     sqlSessionTemplate 相当于我们的SqlSession
45     -->
46     <bean id="sqlSessionTemplate"
class="org.mybatis.spring.SqlSessionTemplate">
47         <!--看源码可知, 这里缺了个构造参数的注入
48         我们只能使用构造器注入, 因为没有set方法 (源码有)
49         -->
50         <constructor-arg index="0" ref="sqlSessionFactory"/>
51     </bean>
52
53
54 </beans>

```

- 因为在spring-dao中有所配置了一些mybatis的配置, 所以我们的myabtis的核心配置文件以后一般只写两种配置:
 - 别名
 - setting

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE configuration
3      PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4      "http://mybatis.org/dtd/mybatis-3-config.dtd">
5
6  <configuration>
7      <typeAliases>
8          <package name="com.kuang.pojo"/>
9      </typeAliases>
10
11
12 </configuration>

```

2.2 需要给接口加一个实现类

```

1  public class UserMapperImpl implements UserMapper {
2      //在原来, 我们所有的操作都是用SqlSession, 而现在使用SqlSessionTemplate
3      private sqlSessionTemplate sqlSessionTemplate;
4
5      //注入SqlSessionTemplate
6      public void setSqlSessionTemplate(SqlSessionTemplate
sqlSessionTemplate) {
7          this.sqlSessionTemplate = sqlSessionTemplate;
8      }

```

```

9
10
11     @Override
12     public List<User> getUsers() {
13
14         UserMapper mapper = sqlSessionTemplate.getMapper(UserMapper.class);
15         return mapper.getUsers();
16     }
17 }

```

2.2.1 这是和mybatis不同的地方：接口有了一个实现类。

2.2.2 这个实现类中我们关注两个地方：

- SqlSessionTemplate：原先我们使用SqlSession，现在我们使用这个进行操作（增删改查），比如getMapper。
- 注入SqlSessionTemplate：记得在applicationContext管理这个接口实现类的时候，记得使用set注入

2.2.3 这个接口实现类有什么意义？

- 完成一些业务操作
- 我们后续只要调用这个接口实现类的方法即可
- 接口实现类记得注册到Spring容器中，让Spring管理

2.3 测试运行

```

1  @Test
2  public void getUsers2(){
3      ApplicationContext context = new
4      ClassPathXmlApplicationContext("applicationContext.xml");
5
6      UserMapper userMapper = context.getBean("userMapper",
7      UserMapper.class);
8      List<User> users = userMapper.getUsers();
9
10     for (User user : users) {
11         System.out.println(user);
12     }
13 }

```

```

User(id=1, name=test, pwd=123)
User(id=3, name=jerry, pwd=456789)
User(id=4, name=李三, pwd=123)
User(id=6, name=李明, pwd=ff3rwf)

Process finished with exit code 0

```

3 SqlSessionDaoSupport（方式二）

SqlSessionDaoSupport是一个抽象的支持类，用来为你提供 SqlSession。调用 `getSqlSession()` 方法你会得到一个 SqlSessionTemplate，之后可以用于执行 SQL 方法。

- 接口实现类

```
1 public class UserMapperImpl2 extends SqlSessionDaoSupport implements
  UserMapper {
2     @Override
3     public List<User> getUsers() {
4         return getSqlSession().getMapper(UserMapper.class).getUsers();
5     }
6 }
```

- 注册bean, 在bean中配置sqlSessionFactory即可

```
1 <bean id="userMapper2" class="com.kuang.mapperImpl.UserMapperImpl2">
2     <property name="sqlSessionFactory" ref="sqlSessionFactory"/>
3 </bean>
```

- 测试运行

```
1 @Test
2 public void getUsersbydaosupport(){
3     ApplicationContext context = new
4     ClassPathXmlApplicationContext("applicationContext.xml");
5     UserMapper userMapper = context.getBean("userMapper2",
6     UserMapper.class);
7     List<User> users = userMapper.getUsers();
8     for (User user : users) {
9         System.out.println(user);
10    }
11 }
```

```
User(id=1, name=test, pwd=123)
User(id=3, name=jerry, pwd=456789)
User(id=4, name=李三, pwd=123)
User(id=6, name=李明, pwd=ff3rwf)

Process finished with exit code 0
```

注意点:

1. bean中需要配置SqlSessionFactory (核心)
2. 可以省略之前配置的sqlSessionTemplate, 因为通过getSession()方法会得到一个SqlSessionTemplate