一.准备工作,需要的架包jar:

1.MyBatis和Spring的组合(mybatis-spring)

2.Spring核心容器(spring-context)

3.Spring对数据库访问的支持(spring-jdbc)

4.面向切面编程(aspectjweaver)

5.Mysql的驱动(mysql-connector-java),当前用的数据库版本相匹配,当前版本:5.0.8

6.数据库连接池(druid)

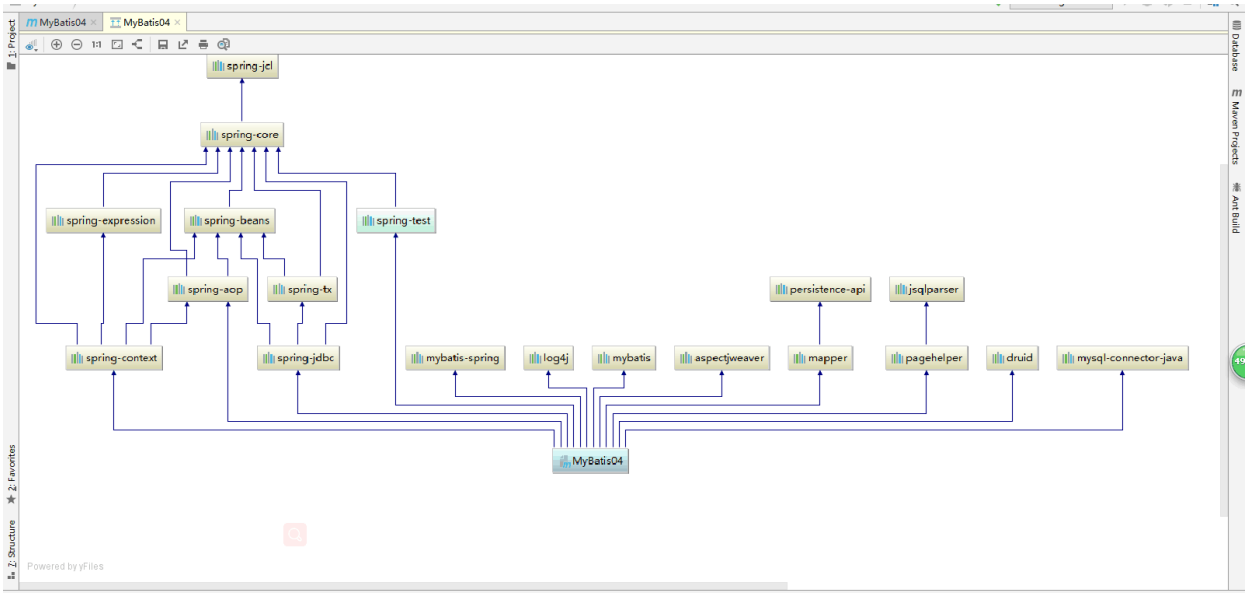
7.Spring对测试框架(spring-test)

8.log4j日志文档(log4j)

9.MyBatis的分页插件(pagehelper)

10.通用mapper(mapper)

11.Junit测试框架(5.2)



二.结合spring-mybatis网站对两个框架学习:http://www.mybatis.org/spring/zh/getting-started.html

1.首先,新建Spring-config.xml,配置Spring,我们通过注解包扫描的方式,进行扫描配置是数据库连接池

```
<!--包扫描-->
<!--<context:component-scan base-package="com.lanou"/>-->
<context:component-scan base-package="com.lanou.dao,com.lanou.service"/>

<!--导入properties文件-->
<context:property-placeholder location="jdbc.properties"/>

<!--配置数据库连接池-->

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
  <property name="driverClassName" value="${jdbc.driver}"/>
  <property name="url" value="${jdbc.url}"/>
  <property name="username" value="${jdbc.username}"/>
  <property name="password" value="${jdbc.password}"/>
</bean>
```

2.其次,在 MyBatis-Spring 中,SqlSessionFactoryBean 是用于创建 SqlSessionFactory 的。要 配置这个工厂 bean,放置下面的代码在 Spring 的 XML 配置文件中:(区别:Spring中我们需要从容器中生成 SqlSessionFactory)

```
<!--SqlSessionFactoryBean用于创建SqlSessionFactory-->
<bean id="SqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"/>
```

注:如果包不在一个路径下,设置MyBatis的配置文件路径,提供了一种简便的方法(不用创建mybatis的配置日志,直接可以用configuration在spring-config中配置)**

```
问题1:设置MyBatis的配置文件路径(可选)
<property name="configLocation" value="mybatis-config.xml"/>
问题2:如果UserDao和UserDao.xml,在同一个路径下,可以自动识别,不在同一个路径系下,
需要设置xml路径
<property name="mapperLocations" value="classpath:com/lanou/dao/*.xml">

```

```
</property>
```

```
<!--相当于Mybatis的配置文件-->
<property name="configuration">
 <bean class="org.apache.ibatis.session.Configuration">
 <property name="logImpl" value="org.apache.ibatis.logging.log4j.Log4jImpl"/>
 //mapUnderscoreToCamelCase:自动将有_转成驼峰命名
 <property name="mapUnderscoreToCamelCase" value="true"/>
 </bean>
</property>
```

三.注入映射关系

扫描dao层的接口的两种方式,推荐使用第二种,(原因:可以扫描多个dao层文件)

```
方式1:<bean id="userDao" class="org.mybatis.spring.mapper.MapperFactoryBean">
<property name="mapperInterface" value="com.lanou.dao.UserDao"/>
<property name="sqlSessionFactory" ref="SqlSessionFactoryBean"/>
</bean>
```

```
方式2:<!--扫描dao层接口-->
<bean class="tk.mybatis.spring.mapper.MapperScannerConfigurer">
 <property name="basePackage" value="com.lanou.dao"/>
</bean>
```

四.事务

开启事务,Spring和mybatis都有开启事务,我们推荐使用spring的开启事务.

```
service业务逻辑层,实现的impl中
1.找到全部只需要调用dao层的selectall即可
2.开始事务会在开启事务注解,在service实现类中会自动去service业务处理层找,
@Transactional@Service到service层,@Autowired自动装配
源代码:
//异常时回滚
@Transactional(rollbackFor = Exception.class)
```

```
@Service
public class UserServiceImpl implements UserService {
  @Autowired
  private UserDao userDao;
  public List<User> findUsers() {
  return userDao.selectAll();


  }

  public PageInfo<User> findUsers(int page, int count) {
  PageHelper.startPage(page,count);
  List<User> userList = userDao.selectAll();
  PageInfo<User> userPageInfo = new PageInfo<User>(userList);
  return userPageInfo;
  }
 }
<!--开启事务-->
<bean id="transactionManager" class="org.springframework.jdbc.da
tasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource"/>
</bean>

<!--aop,方法级别的-->
<!--开启事务注解-->
<tx:annotation-driven transaction-manager="transactionManager"/>

 <!--开启aop的注解-->
 <aop:aspectj-autoproxy/>
```

五.分页查询

对User用户分页查询,在dao层写接口,在xml实现到层的方法(sql语句)service调用dao层的实现.如上,分页查询.网站:https://pagehelper.github.io/学习.导入pagehelper分页插件.

在value中配置可选参数.第一个值是方言sql,分页合理化参数，默认值为`false`。当该参数设置为 `true` 时，`pageNum<=0` 时会查询第一页

2. 在 Spring 配置文件中配置拦截器插件

使用 spring 的属性配置方式，可以使用 `plugins` 属性像下面这样配置：

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <!-- 注意其他配置 -->
  <property name="plugins">
    <array>
      <bean class="com.github.pagehelper.PageInterceptor">
        <property name="properties">
          <!--使用下面的方式配置参数，一行配置一个 -->
          <value>
            params=value1
          </value>
        </property>
      </bean>
    </array>
  </property>
</bean>
```

```
<bean id="SqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"/>

  <!--相当于Mybatis的配置文件-->
  <property name="configuration">
  <bean class="org.apache.ibatis.session.Configuration">
  <property name="logImpl" value="org.apache.ibatis.logging.log4j.Log4jImpl"/>
  <property name="mapUnderscoreToCamelCase" value="true"/>
  </bean>
  </property>
  <!--添加mybatis的分页插件pageHelper-->
  <property name="plugins">
  <array>
  <bean class="com.github.pagehelper.PageInterceptor">
  <property name="properties">
  <!--pageHelper的参数-->
  <value>
  helperDialect=mysql
  reasonable=true
  </value>
  </property>
  </bean>
```

```
 </array>
 </property>
</bean>
```

```
测试:@Test
void test4() {
 PageHelper.startPage(1,2);
 List<User> userList = userDao.selectAll();
 for (User user : userList) {
 System.out.println(user);
 }
}
```

主要使用pagehelper和pageinfo,再次调用Userservice层.

```
@Test
void test5() {
 PageInfo<User> pageInfo = userService.findUsers(1, 2);
 System.out.println(pageInfo);
}
```

输出结果:

```
DEBUG [main] - <==      Total: 2
PageInfo{pageNum=1, pageSize=2, size=2, startRow=1, endRow=2, total=7, pages=4, list=Page{count=true,
 pageNum=1, pageSize=2, startRow=0, endRow=2, total=7, pages=4, reasonable=true,
 pageSizeZero=false}[User{id=1, username='333344', password='9999999'}, User{id=5, username='234rr4',
 password='dffefe34'}], prePage=0, nextPage=2, isFirstPage=true, isLastPage=false,
 hasPreviousPage=false, hasNextPage=true, navigatePages=8, navigateFirstPage=1, navigateLastPage=4,
 navigatepageNums=[1, 2, 3, 4]}
```

```
其中字段代表的意思:
//总记录数
private long total;
//总页数
private int pages;
//结果集
private List<T> list;

//第一页
 private int firstPage;
```

```
//前一页
private int prePage;
//下一页
private int nextPage;
//最后一页
private int lastPage;

//是否为第一页
private boolean isFirstPage = false;
//是否为最后一页
private boolean isLastPage = false;
//是否有前一页
private boolean hasPreviousPage = false;
//是否有下一页
private boolean hasNextPage = false;
//导航页码数
private int navigatePages;
//所有导航页号
private int[] navigatepageNums;
```

六.通用mapper

为什么使用通用mapper?

通用 Mapper4 是一个可以实现任意 MyBatis 通用方法的框架，项目提供了常规的增删改查操作以及Example 相关的单表操作。通用 Mapper 是为了解决 MyBatis 使用中 90% 的基本操作，使用它可以很方便的进行开发，可以节省开发人员大量的时间。

快捷生成一系列除测试类的所有方式的插件:codehelpel,-问题:会自动生成增,删,改,但是不一定全部都需要.我们引入了通用mapper

1.修改包扫描:扫描dao层的接口org改为tx

```
<bean class="tk.mybatis.spring.mapper.MapperScannerConfigurer">
 <property name="basePackage" value="com.lanou.dao"/>
</bean>
```

2.新建学生的pojo,写dao层的方法,继承于Mapper<Student>.Mapper中有对对象的增,删,改操作

```
1
2 @Repository
3 public interface StudentDao extends Mapper<Student> {
4
5 }
```

测试类:先自动装配私有属性,student.其中有很多方法.

```
1 @Test
2 void test6() {
3  List<Student> studentList = studentDao.selectAll();
4  for (Student student : studentList) {
5  System.out.println(student);
6  }
7 }
```