今日知识点：

1. MySQL数据库
2. MyBatis原理
3. MyBatis映射
4. MyBatis-plus

**什么是Mybatis**

MyBatis 是一款优秀的持久层框架，它支持自定义 SQL、存储过程以及高级映射。MyBatis 免除了几乎所有的 JDBC 代码以及设置参数和获取结果集的工作。

SpringBoot整合MyBatis进行使用

步骤1：修改pom.xml-添加jar包依赖

<**dependencies**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
 *<!--引入thymeleaf依赖-->* <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-thymeleaf</**artifactId**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-jdbc</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 <**version**>5.1.10</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.fasterxml.jackson.core</**groupId**>  
 <**artifactId**>jackson-core</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.fasterxml.jackson.core</**groupId**>  
 <**artifactId**>jackson-databind</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.fasterxml.jackson.datatype</**groupId**>  
 <**artifactId**>jackson-datatype-joda</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.fasterxml.jackson.module</**groupId**>  
 <**artifactId**>jackson-module-parameter-names</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.projectlombok</**groupId**>  
 <**artifactId**>lombok</**artifactId**>  
 <**optional**>true</**optional**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.baomidou</**groupId**>  
 <**artifactId**>mybatis-plus-boot-starter</**artifactId**>  
 <**version**>3.3.2</**version**>  
 </**dependency**>  
 *<!-- 分页插件 -->* <**dependency**>  
 <**groupId**>com.github.pagehelper</**groupId**>  
 <**artifactId**>pagehelper-spring-boot-starter</**artifactId**>  
 <**version**>1.2.12</**version**>  
 </**dependency**>  
   
 *<!-- alibaba的druid数据库连接池 -->* <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>druid-spring-boot-starter</**artifactId**>  
 <**version**>1.1.13</**version**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
 <**exclusions**>  
 <**exclusion**>  
 <**groupId**>org.junit.vintage</**groupId**>  
 <**artifactId**>junit-vintage-engine</**artifactId**>  
 </**exclusion**>  
 </**exclusions**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-test</**artifactId**>  
 <**version**>2.3.1.RELEASE</**version**>  
 <**scope**>compile</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>junit</**groupId**>  
 <**artifactId**>junit</**artifactId**>  
 <**version**>4.12</**version**>  
 <**scope**>compile</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-test</**artifactId**>  
 <**version**>5.2.7.RELEASE</**version**>  
 <**scope**>compile</**scope**>  
 </**dependency**>  
</**dependencies**>

步骤2：修改application.yml配置

**server**:  
 **port**: 8083  
**spring**:  
 **thymeleaf**:  
 **enabled**: **true** *#开启thymeleaf视图解析* **encoding**: utf-8 *#编码* **prefix**: classpath:/templates/ *#前缀* **cache**: **false** *#是否使用缓存* **mode**: HTML *#严格的HTML语法模式* **suffix**: .html *#后缀名* **datasource**:  
 **name**: test  
 **url**: jdbc:mysql://127.0.0.1:3306/dfrz\_edusys?useUnicode=true&characterEncoding=utf8  
 **username**: root  
 **password**: root  
 *# 使用druid数据源* **type**: com.alibaba.druid.pool.DruidDataSource  
 **driver-class-name**: com.mysql.jdbc.Driver  
 **filters**: stat  
 **maxActive**: 20  
 **initialSize**: 1  
 **maxWait**: 60000  
 **minIdle**: 1  
 **timeBetweenEvictionRunsMillis**: 60000  
 **minEvictableIdleTimeMillis**: 300000  
 **validationQuery**: select 'x'  
 **testWhileIdle**: true  
 **testOnBorrow**: false  
 **testOnReturn**: false  
 **poolPreparedStatements**: true  
 **maxOpenPreparedStatements**: 20  
  
*## 该配置节点为独立的节点，有很多同学容易将这个配置放在spring的节点下，导致配置无法被识别***mybatis**:  
 **mapper-locations**: classpath:mapping/\*.xml *#注意：一定要对应mapper映射xml文件的所在路径* **type-aliases-package**: com.dfrz.edusys3.bean *# 注意：对应实体类的路径  
#pagehelper分页插件***pagehelper**:  
 **helperDialect**: mysql  
 **reasonable**: true  
 **supportMethodsArguments**: true  
 **params**: count=countSql  
  
*#mybatis-plus***mybatis-plus**:  
 **mapper-locations**: classpath:mapping/\*.xml  
 **configuration**:  
 *#开启MyBatis日志* **log-impl**: org.apache.ibatis.logging.stdout.StdOutImpl  
 **cache-enabled**: **true**

步骤3：修改User类

*/\*用户类\*/*@TableName(**"t\_user"**)  
**public class** User {  
 *//主键* @TableId(value = **"id"**,type = IdType.***AUTO***)  
 **private** Integer **id**;  
 *//用户名* **private** String **uname**;  
 *//密码* **private** String **upass**;  
  
 **public** User() {  
 }  
  
 **public** User(Integer id, String uname, String upass) {  
 **this**.**id** = id;  
 **this**.**uname** = uname;  
 **this**.**upass** = upass;  
 }

Setter/getter.......

步骤4：在mapper包下建立 UserMapper接口

**public interface** UserMapper **extends** BaseMapper<User> {  
}

步骤5：进行单元测试mapper功能

@RunWith(SpringRunner.**class**)  
@SpringBootTest  
**public class** SpringTest2 {  
 @Autowired  
 UserMapper **userMapper**;  
  
 @Test  
 **public void** test01(){  
 User user=**userMapper**.selectById(1);  
 System.***out***.println(user);  
 }  
 @Test  
 **public void** test02(){  
 User user=**userMapper**.selectById(33);  
 System.***out***.println(user);  
 }  
 @Test  
 **public void** test03(){  
 *//添加用户* User user=**new** User(**null**,**"testman20200701"**,**"0701"**);  
 **userMapper**.insert(user);  
 }  
 @Test  
 **public void** test04(){  
 *//查询所有用户* List<User> list= **userMapper**.selectList(**null**);  
 **for** (User u:list  
 ) {  
 System.***out***.println(u);  
 }  
 }  
  
 @Test  
 **public void** test05(){  
 *//带条件查询* QueryWrapper<User> wrapper=**new** QueryWrapper<>();  
 wrapper.eq(**"uname"**,**"testman20200701"**);  
 List<User> list= **userMapper**.selectList(wrapper);  
 **for** (User u:list  
 ) {  
 System.***out***.println(u);  
 }  
 }  
}

完成业务层和数据库mapper层的对接

@Service  
**public class** UserServiceImpl **implements** IUserService {  
 *//传统模式 Service 和 DAO 耦合了  
 //Spring的注入方式就是来解决 组件和组件之间的耦合性问题  
 //@Autowired 自动装配的注入方式 ：寻找实现IUserDao接口的实现类  
  
// @Autowired  
// @Qualifier(value = "mybatis")  
// IUserDao userDao;* @Autowired  
 UserMapper **userMapper**;  
  
 @Override  
 **public void** addUser(User user) {  
 **userMapper**.insert(user);  
 }  
  
 @Override  
 **public** User login(User user) {  
 **if**(user.getUname().equals(**"zhangsan"**) && user.getUpass().equals(**"000"**)){  
 **return** user;  
 }  
 **else**{  
 **return null**;  
 }  
  
 }  
  
 @Override  
 **public** List<User> getUsers() {  
 List<User> list=**userMapper**.selectList(**null**);  
 */\*list.add(new User(1,"zhangsan","333"));  
 list.add(new User(2,"lisi","1234"));  
 list.add(new User(3,"wangwu","555"));  
 list.add(new User(4,"zhaoliu","888"));  
 list.add(new User(5,"zhangqi","777"));\*/* **return** list;  
 }  
}

MyBatis自定义查询

步骤1：编写接口自定义方法

**public interface** UserMapper **extends** BaseMapper<User> {  
  
 *//自定义查询* **public** User getUserById(Integer id);  
}

步骤2：在mapping文件夹建立UserMapper.xml映射文件

*<?***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" *>***<**mapper namespace="com.dfrz.edusys3.mapper.UserMapper"** >  
 *<!--定义结果映射-->* <**resultMap id="UserResultMap" type="com.dfrz.edusys3.bean.User"**>  
 *<!--主键-->* <**id column="id" property="id"**></**id**>  
 *<!--普通字段-->* <**result column="uname" property="uname"**></**result**>  
 <**result column="upass" property="upass"**></**result**>  
 </**resultMap**>  
 *<!--定义查询-->* <**select id="getUserById" resultMap="UserResultMap"**>  
 select a.id,a.uname,a.upass from t\_user a where id=#{id}  
 </**select**>  
</**mapper**>

步骤3：编写单元测试用例

@Test  
**public void** test06(){  
 User user=**userMapper**.getUserById(1);  
 System.***out***.println(user);  
}