Controller层

@RequestMapping("/testQuery")

**public** String testQuery(

@RequestParam(value="username" ,required =**false**)String username,

@RequestParam(value="password" ,required =**false**)String password,

@RequestParam(value="age" ,required =**false**)**int** age) {

Student student=**new** Student();

student.setAge(age);

student.setUsername(username);

student.setPassword(password);

List<Student> list=studentService.selectByStudent(student);

Iterator it=list.iterator();

**while**(it.hasNext()) {

System.***out***.println(it.next().toString());

}

**return** "query1";

}

Service层

**public** **interface** StudentService {

List<Student> selectByStudent(Student student);

}

**public** **class** StudentServiceImpl **implements** StudentService {

@Override

**public** List<Student> selectByStudent(Student student) {

**return** studentMapper.selectByStudent(student);

}

}

Dao层

**public** **interface** StudentMapper {

List<Student> selectByStudent(Student record);

}

StudentMapper.xml

<select id=*"selectByStudent"* resultType=*"com.demo.model.Student"* >

select\*

from student

<where>

<if test=*"username!=''"*>

and username= #{username}

</if>

<if test=*"password!=''"*>

and password= #{password}

</if>

<if test=*"age!=''"*>

and age= #{age}

</if>

</where>

不写SQL语句，用逆向生成的文件，只在controller层和service层写方法

@RequestMapping("/input/queryOrder")

**public** String queryOrder(@RequestParam(value = "batchNumber", required = **false**) String batchNumber,

@RequestParam(value = "ownerName", required = **false**) String ownerName,

@RequestParam(value = "houseId", required = **false**) String houseIdfromjsp,

@RequestParam(value = "createTimefrom", required = **false**) String createTimefrom,

@RequestParam(value = "createTimeto", required = **false**) String createTimeto ) {

OrderExample example=**new** OrderExample();

Criteria criteria =example.createCriteria();

**if**(!StringUtils.*isEmpty*(batchNumber)) {

criteria.andBatchNumberEqualTo(batchNumber);

}

**if**(!StringUtils.*isEmpty*(ownerName)) {

criteria.andOwnerNameEqualTo(ownerName);

}

**if**(!StringUtils.*isEmpty*(houseIdfromjsp)) {

**int** houseId = Integer.*parseInt*(houseIdfromjsp);

criteria.andHouseIdEqualTo(houseId);

}

**if** ((!StringUtils.*isEmpty*(createTimefrom))&&(!StringUtils.*isEmpty*(createTimeto))) {

SimpleDateFormat sdf = **new** SimpleDateFormat("yyyy-MM-DD");

**try** {

Date datefrom = sdf.parse(createTimefrom);

Date dateto = sdf.parse(createTimeto);

criteria.andCreateTimeBetween(datefrom, dateto);

} **catch** (ParseException e) {

e.printStackTrace();

}

}

List<Order> list = orderService.selectByWarehous(example);

Iterator<Order> it = list.iterator();

**while** (it.hasNext()) {

System.***out***.println(it.next().toString());

}

**return** **null**;

}

联合两张表进行查询

// 结合两张表，输出查询结果，先根据输入的三个信息查询order表，得到与goods表对应的订单号，然后根据订单号查询goods表

@RequestMapping("/input/goodsDetail")

**public** String goodsDetail(String batchNumber, String ownerName, String houseid) {

**int** houseId = Integer.*parseInt*(houseid);// 把输入的String类型的houseid转为int型的houseId

OrderExample orderexample = **new** OrderExample();

orderexample.createCriteria().andBatchNumberEqualTo(batchNumber).andOwnerNameEqualTo(ownerName)

.andHouseIdEqualTo(houseId);

List<Order> list = orderService.selectByWarehous(orderexample);//根据输入的三个信息查询order表

**if** (list != **null**) {

**for** (**int** i = 0; i < list.size(); i++) {

Order orderone = list.get(i);

String ordernumber = orderone.getOrderNumber();//获得查询结果的订单号

System.***out***.println(ordernumber);//测试时打印中间结果

GoodsExample goodsexample = **new** GoodsExample();

goodsexample.createCriteria().andOrderNumberEqualTo(ordernumber);

List<Goods> goodslist = goodsService.selectByOrderNumber(goodsexample);//根据获得的订单号，查询goods表

Iterator<Goods> it = goodslist.iterator();//测试时打印输出结果

**while** (it.hasNext()) {

System.***out***.println(it.next().toString());

}

}

} **else** {

System.***out***.println("未找到相应数据");

}

**return** **null**;

}