目录

[**1 实验开始时间**： 2](#_Toc83461149)

[**2 实验结束时间**： 2](#_Toc83461150)

[**3 需求分析：** 2](#_Toc83461151)

[**4 详细设计：** 2](#_Toc83461152)

[**5 开发和运行环境：** 3](#_Toc83461153)

[**6 源代码：** 3](#_Toc83461154)

[**7实验结果说明：** 13](#_Toc83461155)

[**8 实验问题** 16](#_Toc83461159)

[**9 实验总结：** 16](#_Toc83461160)

**1 实验开始时间**：

2021年9月9日

**2 实验结束时间**：

2021年9月21日

**3 需求分析：**

根据实验要求，本项目需要实现对药品入库单的增删改（查选做）。经过分析，药品入库单中的信息可以大体分为四个实体，分别是入库单单头、药品和供应商、入库单单目。

所以本项目可以具体为根据四者之间的联系，联合对入库单单头、药品、供应商、入库单单目实现增删改查。

所以需要建立四个实体类对应四张表。但同时为了方便保存药品入库单的单目信息，又建立了一个dto（数据传输对象）类ReceiptItem存放药品入库单单目中每一行的信息；建立一个dto类DrugWarehousingForm存放完整的一张入库单的所有信息。

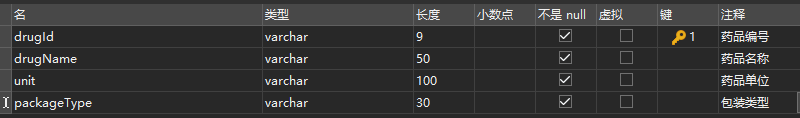
**4 详细设计：**

E-R图：



数据库设计：

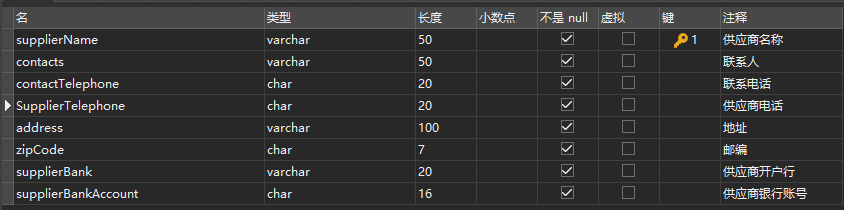
药品表：



入库单单头：



供应商：



入库单单目：

**5 开发和运行环境：**

开发环境：IDEA 2020.2

数据库： MySQL

运行环境：Windows10系统

编程语言：Java

系统架构：SSM

**6 源代码：**

1. 实现数据库操作的主要代码：

药品的增删改查

<insert id="insertDrug">

insert into softwareanalysis.drugdata (drugId, drugName, unit, packageType)

values (#{drugId}, #{drugName}, #{unit}, #{packageType})

</insert>

<select id="queryDrugById" resultType="org.example.domain.Drug">

select \*

from softwareanalysis.drugdata

where drugId = #{drugId}

</select>

<delete id="updateDrug">

update softwareanalysis.drugdata

set drugName = #{drugName},

unit = #{unit},

packageType = #{packageType}

where drugId = #{drugId};

</delete>

<delete id="deleteDrug">

delete

from softwareanalysis.drugdata

where drugId = #{drugId};

</delete>

<select id="queryAllDrugs" resultType="org.example.domain.Drug">

select \*

from softwareanalysis.drugdata

</select>

供应商增删改查

<insert id="insertSupplier">

insert into softwareanalysis.supplierdata (supplierName, contacts, contactTelephone,

SupplierTelephone, address, zipCode, supplierBank, supplierBankAccount)

values (#{supplierName}, #{contacts}, #{contactTelephone}, #{SupplierTelephone},

#{address}, #{zipCode}, #{supplierBank}, #{supplierBankAccount})

</insert>

<select id="querySupplierByName" resultType="org.example.domain.Supplier">

select \*

from softwareanalysis.supplierdata

where supplierName = #{supplierName}

</select>

<delete id="deleteSupplierByName">

delete

from softwareanalysis.supplierdata

where supplierName like #{supplierName}

</delete>

<select id="queryAllSuppliers" resultType="org.example.domain.Supplier">

select \*

from softwareanalysis.supplierdata

</select>

<delete id="updateSupplier">

update softwareanalysis.supplierdata

set contacts =#{contacts},

contactTelephone=#{contactTelephone},

SupplierTelephone=#{SupplierTelephone},

address=#{address},

zipCode=#{zipCode},

supplierBank=#{supplierBank},

supplierBankAccount=#{supplierBankAccount}

where supplierName = #{supplierName};

</delete>

入库单单头增删改查

<insert id="insertHeader">

insert into softwareanalysis.header (warehousingId, indate, indatePerson, auditDate, reviewer, supplierNameRef)

values (#{warehousingId}, #{indate}, #{indatePerson}, #{auditDate}, #{reviewer}, #{supplierNameRef})

</insert>

<select id="selectAllHeader" resultType="org.example.domain.Header">

select \*

from softwareanalysis.header

</select>

<delete id="deleteHeader">

delete

from softwareanalysis.header

where warehousingId = #{warehousingId}

</delete>

<delete id="updateHeader">

update softwareanalysis.header

set indate=#{indate},

indatePerson=#{indatePerson},

auditDate=#{auditDate},

reviewer=#{reviewer},

supplierNameRef=#{supplierNameRef}

where warehousingId = #{warehousingId}

</delete>

<select id="selectHeader" resultType="org.example.domain.Header">

select \*

from softwareanalysis.header

where warehousingId = #{warehousingId}

</select>

入库单单目增删改查

<insert id="insertDetails">

insert into softwareanalysis.receiptitem (orderNumber, purchase, amount, effectiveDate, drugId, warehousingId)

values (#{orderNumber}, #{purchase}, #{amount}, #{effectiveDate}, #{drugId}, #{warehousingId})

</insert>

<select id="queryDetailsByHeaderId" resultType="org.example.domain.Details">

select \*

from softwareanalysis.receiptitem

where warehousingId = #{warehousingId}

</select>

<delete id="updateDetails">

update softwareanalysis.receiptitem

set orderNumber = #{orderNumber},

purchase = #{purchase},

amount = #{amount},

effectiveDate = #{effectiveDate},

drugId = #{drugId},

warehousingId = #{warehousingId}

</delete>

<delete id="deleteDetails">

delete

from softwareanalysis.receiptitem

where warehousingId = #{warehousingId}

</delete>

1. controller的请求处理代码

药品：

// 添加药品请求

@RequestMapping("/addDrug.do")

public ModelAndView addDrug(Drug drug) {

ModelAndView mav = new ModelAndView();

try{

drugService.addDrug(drug);

} catch (Exception e){

mav.addObject("msg", "重复添加药品");

}

mav.setViewName("forward:queryAllDrugs.do");

return mav;

}

// 修改药品请求

@RequestMapping("/modifyDrug.do")

public String modifyDrug(Drug drug) {

drugService.modifyDrug(drug);

return "forward:queryAllDrugs.do";

}

// 查询所有药品请求

@RequestMapping("/queryAllDrugs.do")

public ModelAndView queryAllDrugs() {

ModelAndView mav = new ModelAndView();

List<Drug> drugList = drugService.queryAllDrugs();

mav.addObject("drugList", drugList);

mav.setViewName("viewAllDrugs");

return mav;

}

// 根据id查询药品请求

@RequestMapping("/queryDrugById.do")

public ModelAndView queryDrugById(String drugId) {

ModelAndView mav = new ModelAndView();

Drug drug = drugService.queryDrugById(drugId);

if (drug == null) {

mav.setViewName("forward:queryAllDrugs.do");

mav.addObject("msg", "未查询到药品信息");

return mav;

}

mav.addObject("drug", drug);

mav.setViewName("modifyDrug");

return mav;

}

/\*\*

\* 处理前台ajax请求，实现输入药品id后局部刷新页面，自动显示药品其余信息

\*

\* @param drugId id

\* @return 查到的drug

\*/

@RequestMapping("/queryDrugByIdAjax.do")

@ResponseBody

public Drug queryDrugByIdAjax(String drugId, Integer rows) {

return drugService.queryDrugById(drugId);

}

// 删除药品

@RequestMapping("/deleteDrugById.do")

public String deleteDrugById(String drugId) {

drugService.deleteDrugById(drugId);

return "forward:queryAllDrugs.do";

}

供应商

// 根据名字查询供应商

@RequestMapping("/querySupplierByName.do")

public ModelAndView querySupplierByName(String supplierName) {

ModelAndView mav = new ModelAndView();

Supplier supplier = supplierService.querySupplierByName(supplierName);

if (supplier == null) {

mav.setViewName("forward:queryAllSuppliers.do");

mav.addObject("msg", "未查询到供应商信息");

return mav;

}

mav.addObject("supplier", supplier);

mav.setViewName("modifySupplier");

return mav;

}

/\*\*

\* 处理前端ajax请求，返回supplier对象

\*

\* @param supplierName 前端传近来的要查询的供应商名字

\* @return 返回查询到的供应商信息

\*/

@RequestMapping("/querySupplierByNameAjax.do")

@ResponseBody

public Supplier querySupplierByNameAjax(String supplierName) {

Supplier supplier = supplierService.querySupplierByName(supplierName);

return supplier;

}

// 查询所有供应商

@RequestMapping("/queryAllSuppliers.do")

public ModelAndView queryAllSuppliers() {

ModelAndView mav = new ModelAndView();

List<Supplier> supplierList = supplierService.queryAllSuppliers();

mav.addObject("supplierList", supplierList);

mav.setViewName("viewAllSuppliers");

return mav;

}

// 增加供应商

@RequestMapping("/addSupplier.do")

public ModelAndView addSupplier(Supplier supplier) {

ModelAndView mav = new ModelAndView();

try {

supplierService.addSupplier(supplier);

} catch (Exception e){

mav.addObject("msg", "重复添加供应商");

}

mav.setViewName("forward:queryAllSuppliers.do");

return mav;

}

// 删除供应商

@RequestMapping("/deleteSupplierByName.do")

public String deleteSupplierByName(String supplierName) {

supplierService.deleteSupplierByName(supplierName);

return "forward:queryAllSuppliers.do";

}

// 修改供应商

@RequestMapping("/modifySupplierByName.do")

public String modifySupplierByName(Supplier supplier) {

supplierService.modifySupplier(supplier);

return "forward:queryAllSuppliers.do";

}

入库单：

@RequestMapping("/queryAllForm.do")

public ModelAndView queryAllOrder() {

ModelAndView mav = new ModelAndView();

List<Header> headerList = headerService.queryAllHeader();

mav.addObject("headerList", headerList);

mav.setViewName("viewAllForms");

return mav;

}

@RequestMapping("/queryFormById.do")

public ModelAndView queryFormById(String warehousingId) {

ModelAndView mav = new ModelAndView();

DrugWarehousingForm dwf = new DrugWarehousingForm();

Header header = headerService.queryHeaderById(warehousingId);

if (header == null) {

mav.setViewName("forward:queryAllForm.do");

mav.addObject("msg", "未查询到改入库单");

return mav;

}

dwf.setHeader(header);

Supplier supplier = supplierService.querySupplierByName(header.getSupplierNameRef());

dwf.setSupplier(supplier);

List<Drug> drugList = new ArrayList<>();

List<Details> detailsList = detailsService.queryDetailsByHeaderId(header.getWarehousingId());

for (Details details : detailsList) {

Drug drug = drugService.queryDrugById(details.getDrugId());

drugList.add(drug);

}

dwf.setDetails(detailsList);

dwf.setDrugList(drugList);

mav.addObject("dwf", dwf);

mav.setViewName("modifyForm");

return mav;

}

@RequestMapping("/modifyForm.do")

public String modifyForm(Header header, Supplier supplier, HttpServletRequest request) {

// 获取前端药品资料和入库单单目数据 [{name:zhangsan,id:1001,age:18}，{name:zhangsan,id:1001,age:18}]

String tableList = request.getParameter("tableList");

// 清空之前该入库单下的所有旧数据

detailsService.deleteDetails(header.getWarehousingId());

// 原始数据为字符串[{},{}],在这里先取出最外层的[],然后根据 },{ 分割

String[] detailsStr = StrUtil.removeAll(StrUtil.removeAll(tableList, "["), "]").split("},\\{");

// 对每个元素再进行分割{name:zhangsan,id:1001,age:18}

for (String aa : detailsStr) {

// 将每一条数据添加到集合中，然后挑选出对应信息放到对象中（drug和details是混合在一起的）

List<String> strings = new ArrayList<>();

// 去除"" 和 \

String[] split = StrUtil.removeAll(StrUtil.removeAll(StrUtil.removeAll(aa, "\\"), "\""), "}").split(",");

for (int i = 0; i < split.length; i++) {

// 拼接出一个完整和

strings.add(split[i].split(":")[1]);

}

Details details = new Details(strings.get(0), Convert.toDouble(strings.get(5)),

Convert.toInt(strings.get(6)), strings.get(8), header.getWarehousingId(), strings.get(1));

try {

// 添加新的数据

detailsService.addDetails(details);

} catch (Exception e) {

System.out.println("重复插入");

}

}

header.setSupplierNameRef(supplier.getSupplierName());

headerService.modifyHeader(header);

return "forward:queryAllForm.do";

}

@RequestMapping("/deleteForm.do")

public String deleteForm(String warehousingId) {

headerService.deleteHeader(warehousingId);

detailsService.deleteDetails(warehousingId);

return "forward:queryAllForm.do";

}

@RequestMapping("/addForm.do")

public String addOrder(Header header, Supplier supplier, HttpServletRequest request) {

// 获取前端药品资料和入库单单目数据 [{name:zhangsan,id:1001,age:18}，{name:zhangsan,id:1001,age:18}]

String tableList = request.getParameter("tableList");

// 原始数据为字符串[{},{}],在这里先取出最外层的[],然后根据 },{ 分割

String[] detailsStr = StrUtil.removeAll(StrUtil.removeAll(tableList, "["), "]").split("},\\{");

// 对每个元素再进行分割{name:zhangsan,id:1001,age:18}

for (String aa : detailsStr) {

// 将每一条数据添加到集合中，然后挑选出对应信息放到对象中（drug和details是混合在一起的）

List<String> strings = new ArrayList<>();

// 去除"" 和 \

String[] split = StrUtil.removeAll(StrUtil.removeAll(StrUtil.removeAll(aa, "\\"), "\""), "}").split(",");

for (int i = 0; i < split.length; i++) {

// 拼接出一个完整和

strings.add(split[i].split(":")[1]);

}

Details details = new Details(strings.get(0), Convert.toDouble(strings.get(5)),

Convert.toInt(strings.get(6)), strings.get(8), header.getWarehousingId(), strings.get(1));

try {

detailsService.addDetails(details);

} catch (Exception e) {

System.out.println("重复插入");

e.printStackTrace();

}

}

header.setSupplierNameRef(supplier.getSupplierName());

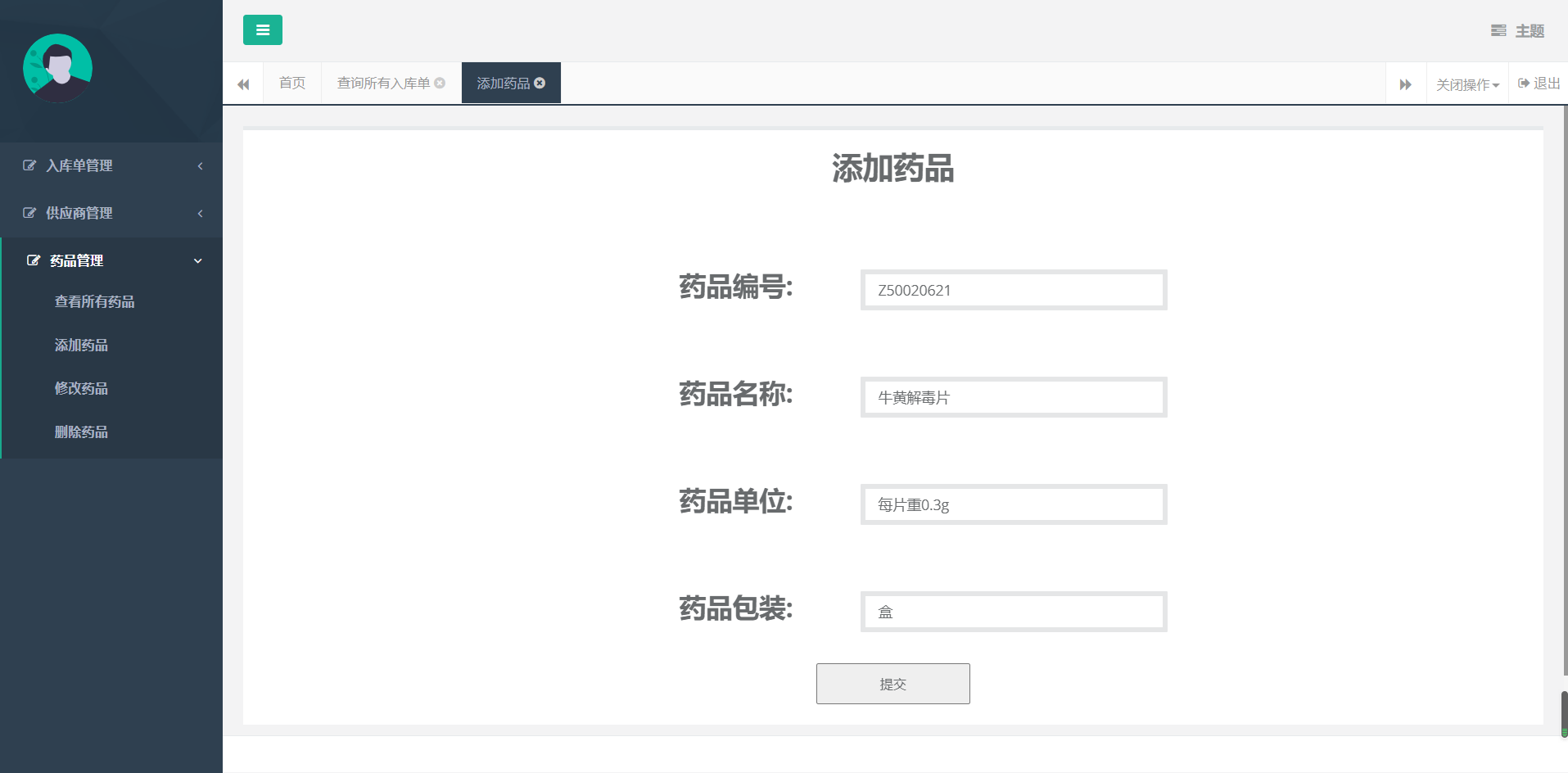
headerService.addHeader(header);

// 转发到查询所有入库单请求

return "forward:queryAllForm.do";

}

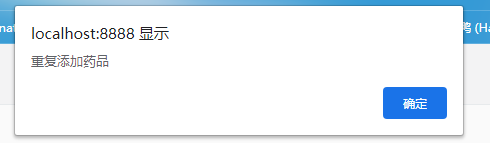
**7实验结果说明：**

**药品添加**

添加成功和失败都会进入查看所有药品界面，但失败会显示提示信息。

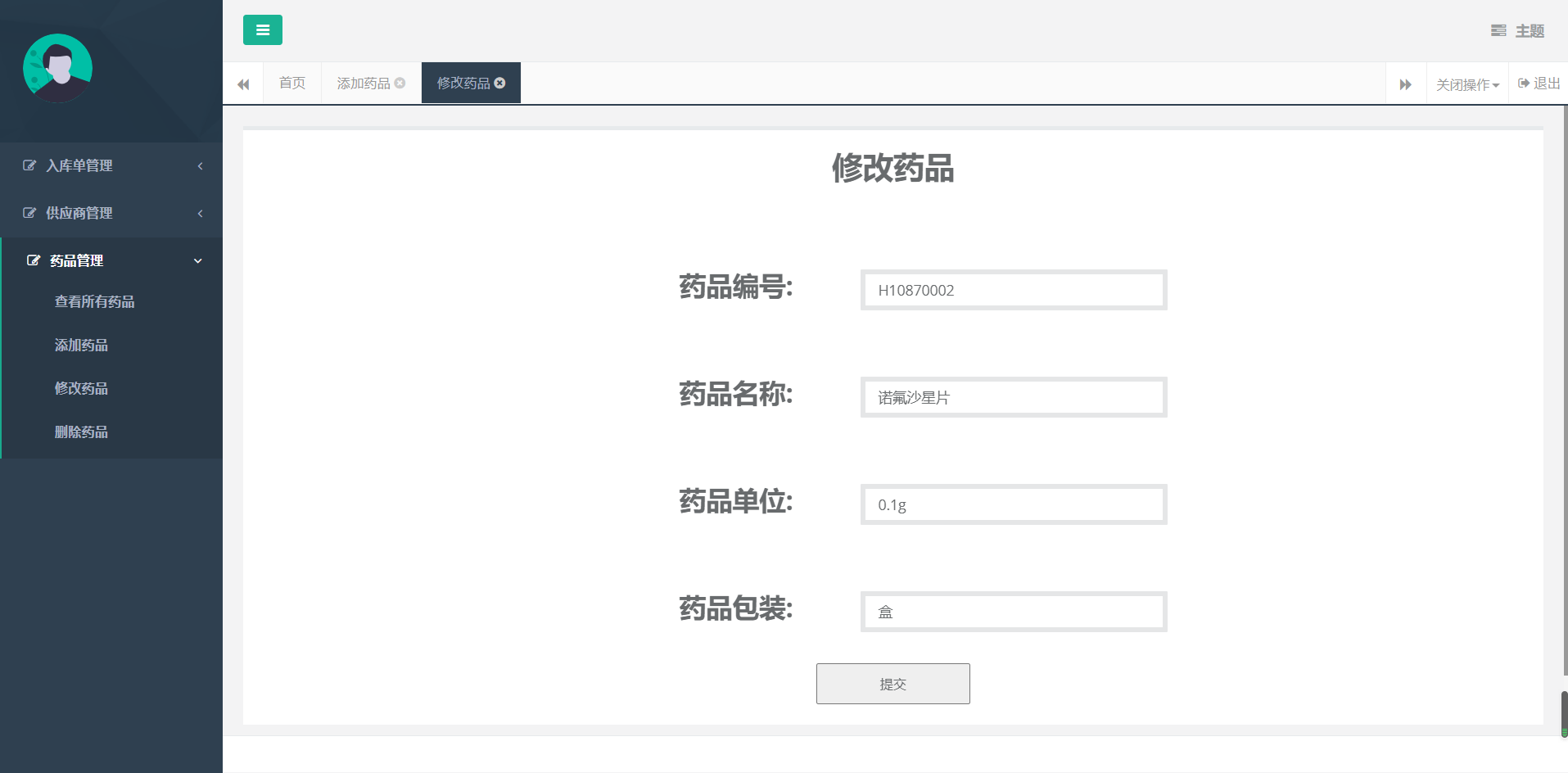
查看所有药品界面可以根据需要点击下方下拉菜单选择每页显示的药品数量，也可以跳转到相应的页面，使用方便。





删除药品，在查看药品界面直接点击右边删除即可一键删除。

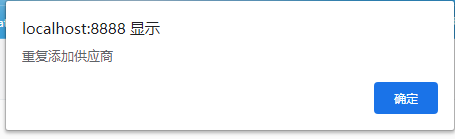
修改药品，在查看药品界面直接点击右边编辑即可进入编辑界面，编辑完成后点击提交即可。



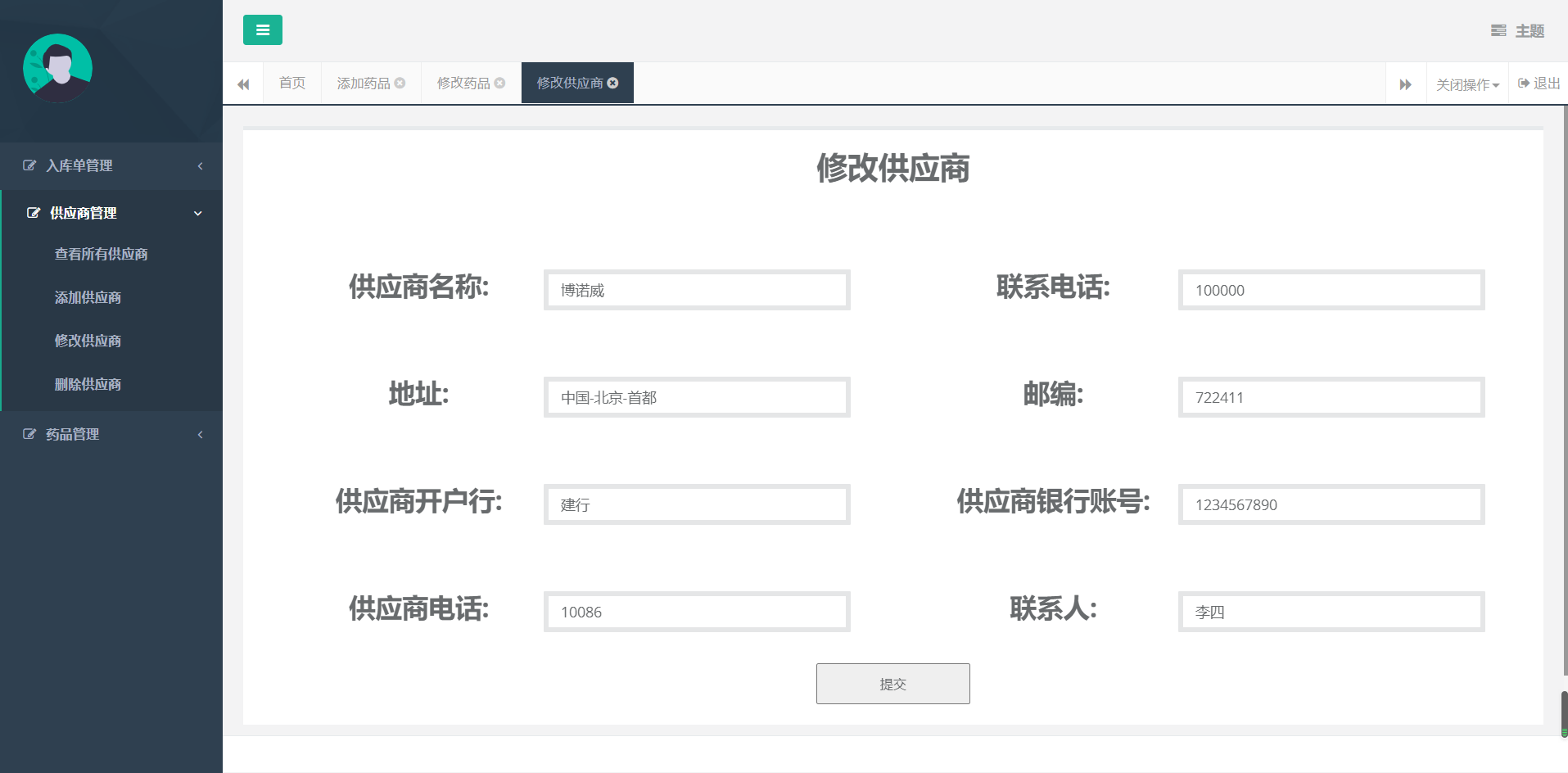
添加供应商：

添加成功和失败都会进入查看所有供应商界面，但失败会显示提示信息





删除供应商，在查看供应商界面直接点击右边删除即可一键删除。

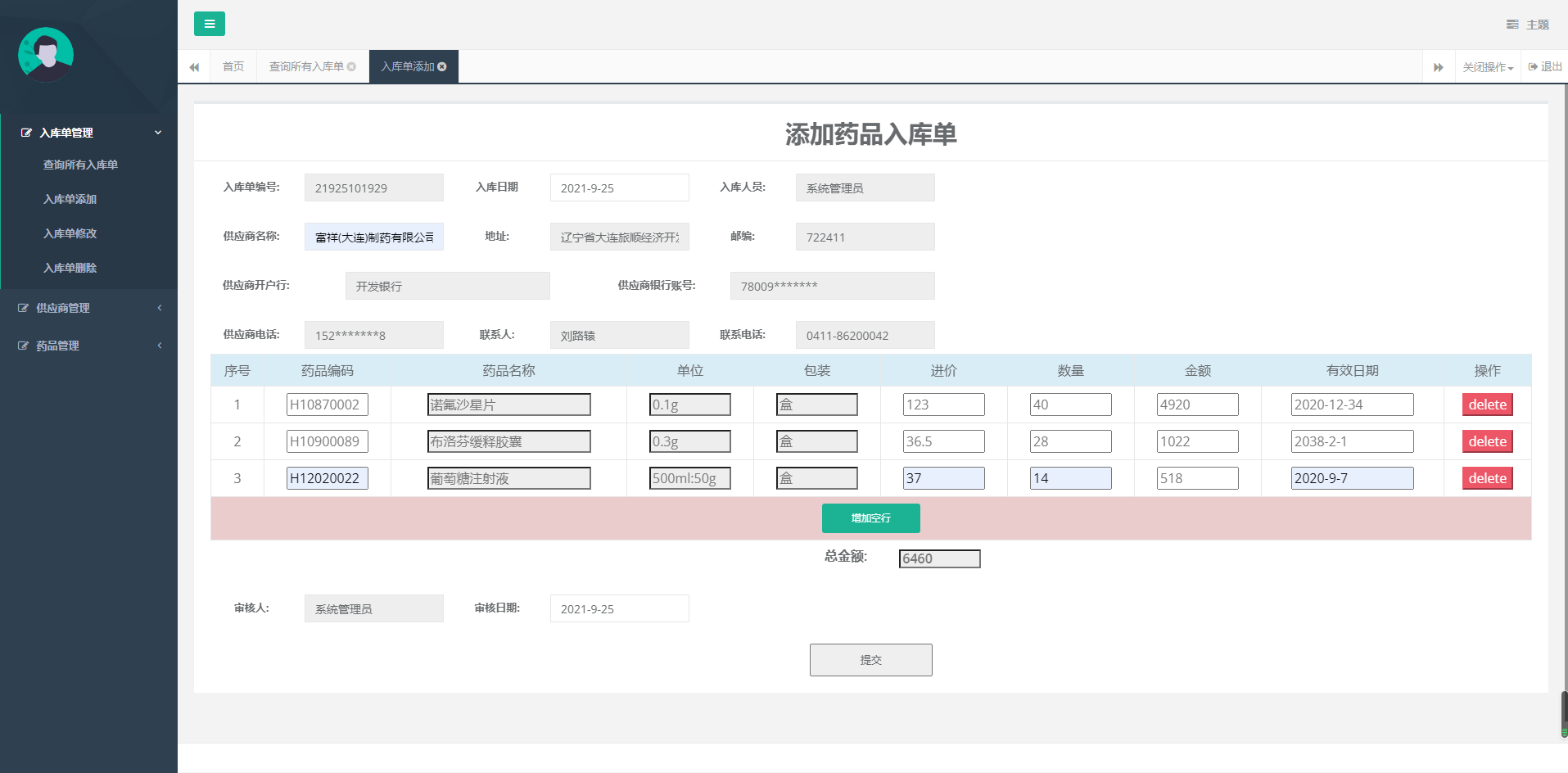
修改供应商，在查看供应商界面直接点击右边编辑即可进入编辑界面，编辑完成后点击提交即可。

添加药品入库单：

灰色输入框均为不可编辑，入库单编号为系统自动生成，入库人员默认为系统最高管理员，入库日期和审核日期均为当前系统日期可编辑。

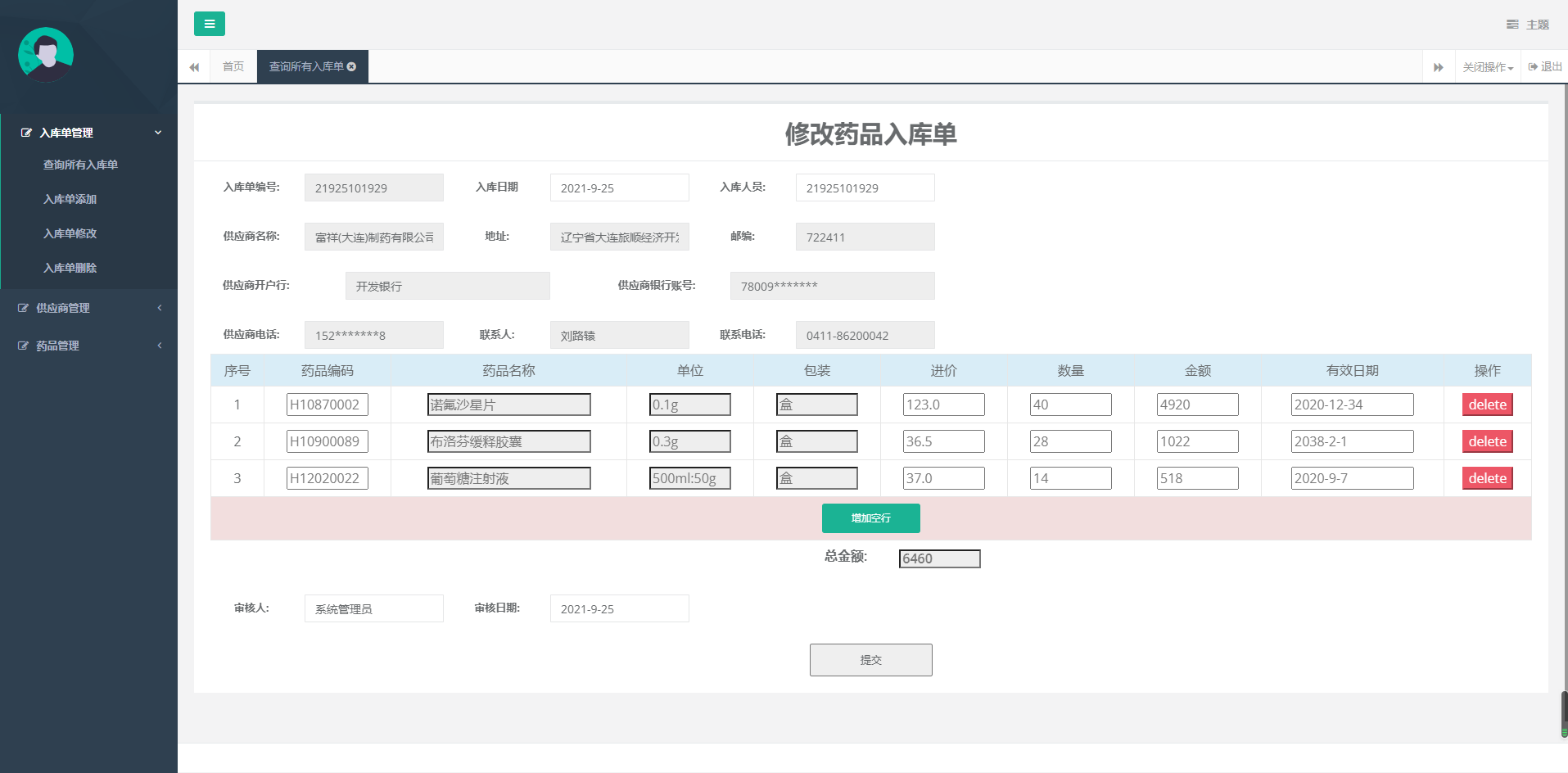
输入供应商名称后其余供应商信息会自动补全；输入药品编码后其余药品信息会自动补全；输入药品的金额和数量后金额会自动计算，并且下方的总金额也会自动更新，极大的方便了用户操作。

入库单单目为响应式表格，可根据需要增加或者删除行，使用方便。



修改入库单。

修改入库单，在查看入库单界面直接点击右边编辑即可进入编辑界面，编辑完成后点击提交即可。



**8 实验问题**

（1）如何从四个表中获取出一个完整的订单信息？ 并赋值给vo类DrugWarehousingForm

解决方法：

a、先从header表中 查询入库单编号得到一个完整的Header ，赋值给 DrugWarehousingForm

b、然后根据header的入库单编号得到供应商信息，赋值给 DrugWarehousingForm

c、然后查询 receiptitem（details） 中所有外键入库单编号为 header的入库单编号的，得到 List<Details>

d、将List<Details>中所有 Details 的 药品编号拿来查询 drugdata 得到 List<Drug>

（2）入库单单目的数据如何临时保存并发送给controller，并且在删除这一行数据时进行页面局部刷新

解决办法：入库单单目的每一行都是利用JavaScript函数进行动态拼接出来的，当添加一条信息都临时以json字符串的方式存储在浏览器的sessionStorage中的变量tableList中，当添加新的一行时在其后边拼接一个空的对象即可；点击删除时删除这一行对应的对象即可，页面则重新利用函数遍历tableList中存储数据的对象，重新显示在页面中。

在点击提交时将其取出赋值给隐藏表单即可将数据发送给后台controller。

（3）输入供应商名称后供应商其余信息自动补全，输入药品编号后药品信息自动补全

解决办法：给供应商名称和药品编码输入框添加键盘监听事件onkeydown和onkeyup，使其调用函数利用ajax请求异步刷新页面

**9 实验总结：**

通过本次实验，让我更加熟悉了软件开发的过程和技术，在项目初期，通过对实验的需求分析，画出了项目的e-r图，并设计了数据库的各个表单和属性以及它们之间的联系，为项目的搭建奠定了基础。

在代码编写过程中，困扰我最久的问题是前端响应式表格的处理，但最终还是成功的解决了这个问题。项目整体功能和使用体验都符合项目的需求，并且实现了选做功能——查询入库单。

此次做项目经历让我学会了处理一对多表单问题，思路更加的清晰与成熟，并且加强了我个人对于JavaScript以及jquery的理解与掌握，在以后的实验中我会更加努力。