项目的第四天 用户认证apache POI导入导出

课程计划：

用户认证：

用户登陆，用户身份认证

药品目录导入（掌握）

药品目录导出（掌握）

# 复习

## 工程聚合

父工程和子工程的pom.xml配置正确，且仓库位置正确，一般不会有问题。

统一版本Maven3.2.1。

父工程的pom.xml配置：

配置子工程中pom.xml

使用相对路径，找子工程中的pom.xml

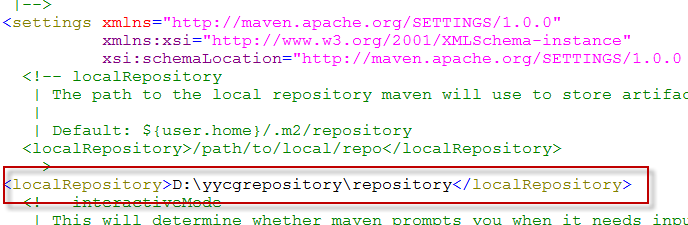


子工程配置：

通过相对路径找父工程中pom.xml



在cmd下聚合，找的全局setting.xml（maven3.2.1安装目录 conf/setting.xml）



如果仓库中没有jar包，联网自动下载，如果网上也没有，需要手动导入jar到仓库。

参考：



将多个子工程聚合到一个war包中，创建聚合工程负责将多个子工程聚合。

工具模块

编译生成jar、发布到仓库

base模块

pom.xml依赖工具jar

技术架构模块

聚合工程

如果没有聚合工程，工程师需要人为判断工程之间依赖关系，按照依赖关系进行单个工程编译、发布。

如果有聚合工程，在聚合工程创建一个聚合的pom.xml（聚合工程只需要这个pom.xml），进入聚合工程目录（目录下一定要有pom.xml），执行maven install。

具体操作过程：

第一步：从svn下得各各工程的代码

第二步：进入聚合工程目录，执行聚合

第三步：将聚合生成的war包拷贝到tomcat。

以上步骤可以通过第三方软件自动化完成，比如：hudson。

# 用户认证

## 什么是用户认证

用户身份认证，是要解决这样的问题：用户告诉系统“我是谁”，系统就问用户凭什么证明你就是“谁”呢？对于采用用户名、密码验证的系统，那么就是出示密码。当用户名和密码匹配，则证明当前用户是谁；对于采用[指纹](http://baike.baidu.com/view/5628.htm)等系统，则出示指纹；对于硬件Key等刷卡系统，则需要刷卡。

静态密码

用户名对应的密码设置在系统，以设置后一般不再改变，安全性低容易被木马窃取，可以定期修改密码，但不容易记忆。

动态密码

现在最常见的短信密码就是动态密码的一种方式。每次都生成一个动态密码，安全性较高。

验证码

验证码不是用户认证的方式，只是通常在用户登录时使用验证码方式。

是一种区分用户是计算机还是人的公共全自动[程序](http://baike.baidu.com/view/17674.htm)。可以防止：恶意破解密码、[刷票](http://baike.baidu.com/view/360266.htm)、论坛灌水，有效防止某个黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试，实际上用验证码是现在很多网站通行的方式（比如招商银行的网上个人银行，百度社区），我们利用比较简易的方式实现了这个功能。

## 用户授权

给用户分配操作权限，用户认证通过后，操作指定权限的功能。

用户授权通过数据模型：

用户表、角色表、权限表(细化到操作链接)、用户角色关系表、角色权限关系表。

企业中在实现用户授权功能时，通常对上边通过数据模型进行修改。

本系统对权限表进行扩展：菜单表（模块表）、操作表（操作链接）

## 用户认证实现流程

**本系统采用用户名密码认证方法。**

第一步：用户访问系统

第二步：系统对用户访问资源校验，该用户是否通过认证

如果用户已经认证，放行，用户继续操作。

如果用户没有认证，如果访问的是公开资源地址（无需认证即可访问），系统就放行，用户继续操作

如果用户没有认证，如果访问的不是公开资源地址，拦截，跳转登陆页面

第三步：用户进入登陆页面

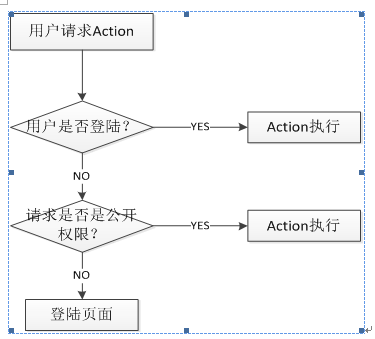
如果用户没有认证，如果访问的不是公开资源地址，拦截，跳转登陆页面

第四步：用户输入用户名、密码进行身份认证

如果认证通过进入系统首页

如果认证不通过进入登陆页面。

认证过程：根据用户账号和密码进行认证。



## 用户认证实现

用户登陆功能实现。

登陆页面上要有验证码：随机产生代码，将代码存入session。

validatecode.jsp(验证码jsp)

随机产生代码存入session：



Service：

接口功能：校验用户身份信息

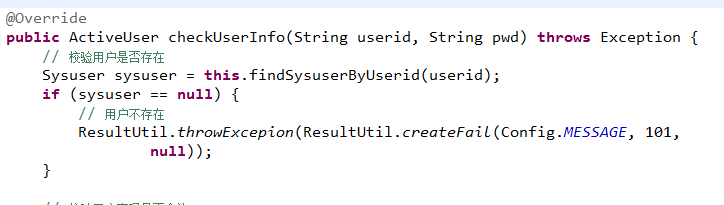
接口参数：用户账号、用户密码

接口返回值：用户身份信息

定义activeuser作为用户身份信息



校验用户身份信息service：



Action :

// 用户登陆提交

@RequestMapping("/loginsubmit")

**public** @ResponseBody

SubmitResultInfo loginsubmit(HttpSession session, String userid, String pwd,String validateCode)

**throws** Exception {

//校验验证码

String validateCode\_session = (String) session.getAttribute("validateCode");

**if**(validateCode\_session!=**null** && !validateCode\_session.equals(validateCode)){

//验证码输入错误

ResultUtil.*throwExcepion*(ResultUtil.*createFail*(Config.*MESSAGE*, 113,

**null**));

}

// service，用户认证

ActiveUser activeUser = userService.checkUserInfo(userid, pwd);

// 将用户身份信息写入session

session.setAttribute(Config.*ACTIVEUSER\_KEY*, activeUser);

**return** ResultUtil.*createSubmitResult*(ResultUtil.*createSuccess*(

Config.*MESSAGE*, 107, **new** Object[] { "" }));

}

页面：

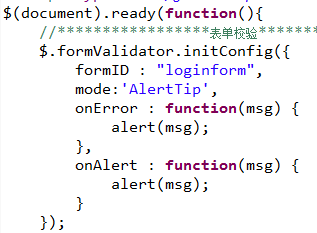
login.jsp

使用AJAX的form提交用户账号和密码及验证码



关于表单校验：

在预加载方法设置表单的校验规则：

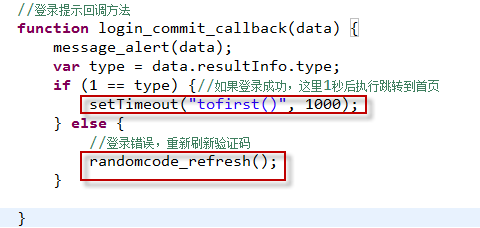


详细参考下边的目录中demo。



如果登陆成功跳转首页

否则刷新验证码。



## 用户身份校验

通常使用filter过虑 器，本系统采用springmvc拦截器实现。

springmvc拦截器是基于处理器映射器HandlerMapping进行拦截。

可以采用一种方法定义一处拦截器，自动在每个HandlerMapping中进行注册。



拦截器定义，需要实现HanlderInterceptor。

配置公开资源 地址

配置文件中配置公开访问地址：



取出该配置文件中公开 地址：

使用工具类，ResourcesUtil中的gekeyList方法

用户身份校验拦截器

**public** **class** LoginInterceptor **implements** HandlerInterceptor{

//执行时机：进入action方法之前执行

//使用场景：用于用户认证、用户授权拦截

@Override

**public** **boolean** preHandle(HttpServletRequest request,

HttpServletResponse response, Object handler) **throws** Exception {

//校验用户身份是否合法

HttpSession session = request.getSession();

ActiveUser activeUser = (ActiveUser) session.getAttribute(Config.*ACTIVEUSER\_KEY*);

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

//用户已经登陆，放行

**return** **true**;

}

//校验用户访问是否是公开资源 地址

List<String> open\_urls = ResourcesUtil.*gekeyList*(Config.*ANONYMOUS\_ACTIONS*);

//用户访问的url

String url = request.getRequestURI();

**for**(String open\_url:open\_urls){

**if**(url.indexOf(open\_url)>=0){

//如果访问的是公开 地址则放行

**return** **true**;

}

}

//拦截用户操作，跳转到登陆页面

request.getRequestDispatcher("/WEB-INF/jsp/base/login.jsp").forward(request, response);

**return** **false**;

}

//执行时机：进入action方法，在返回modelAndView之前执行

//使用场景：在这里统一对返回数据进行处理，比如统一添加菜单 导航

@Override

**public** **void** postHandle(HttpServletRequest request,

HttpServletResponse response, Object handler,

ModelAndView modelAndView) **throws** Exception {

}

//执行时机:action方法执行完成，已经 返回modelAndView，执行。

//使用场景：统一处理系统异常，在这里统一记录系统日志 ，监控action方法执行时间，在preHandle记录开始时间，在afterCompletion记录结束时间

@Override

**public** **void** afterCompletion(HttpServletRequest request,

HttpServletResponse response, Object handler, Exception ex)

**throws** Exception {

}

}

## 测试用户身份校验

1、请求系统首先（非公开地址）

自动跳转到登陆页面。

用户登陆，拦截器放行。

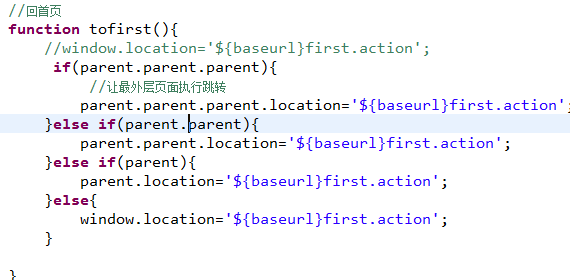
2、在嵌入的页面中，如果session过期了，跳转到登陆页面

现象：

登陆页面是嵌入在页面中。



修改登陆提交的回调方法，



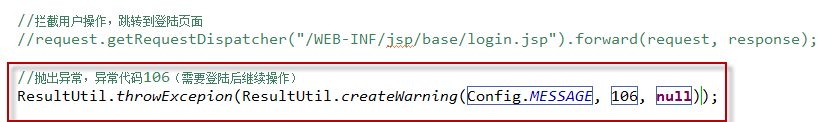
3、当执行一个提交操作时，session过期，应该跳转登陆页面

本系统提交返回是json(SubmitResultInfo) ,由于LoginInterceptor拦截器作用，拦截了，返回了登陆页面。

无法解析登陆页面内容。

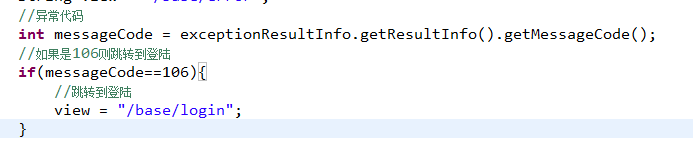
解决：

在LoginInterceptor中，对需要跳转登陆页面，抛出106异常。



在全局异常处理器解析106异常。

解析异常代码是否是106如果是106跳转登陆页面



# 药品目录接口

省级药品采购平台

药品中标目录 ：（药品通用名、药品规格、。。中标价）

药品目录 接口

市级药品采购平台

药品中标目录 ：（药品通用名、药品规格、。。中标价）

在市级药品采购平台上，开发一个药品目录 导入、导出功能。

用户导入过程：

用户线下整理excel文件（记录要导入药品目录 信息）

注意：excel文件规则（定义了文件内容格式，比如第一列是药品流水号、第二列是药品名称。。。。）

进入市级药品采购平台，使用药品目录 导入功能，将线下excel文件导入平台上。

系统按照上边excel文件规则进行文件内容解析，插入到系统数据库。

导入功能好处：

方便执行批量操作

对于复杂的信息，可以采用导入功能，由用户在线下整理内容，导入系统。效率是很快。

导出功能好处：

方便用户备份重要数据信息。

将数据导出到本地，方便对数据进行二次加工。

**建议：所有统计功能，都要添加导出功能。**

## 药品目录 表

系统数据库中，创建一个张表ypxx（药品目录表）：

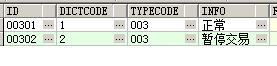
记录了市级医药平台所使用的所有药品（平台药品总目录）

药品流水号：

省平台和市平台统一使用的号，此流水号在省级招标平台产生

药品通用名：（行业内对药品的通用称呼）、剂型、规格、转换系数、生产企业、商品名、药品中标价

药品交易状态（业务代码：1：正常、2：暂停）



唯一约束：



系统中不对药品目录 中的记录进行物理删除，如果这个药品不在系统中使用，设置交易状态为2。

# 药品目录导出

## 导出流程

第一步；根据查询条件(导出条件)查询药品目录表

第二步：将查询到药品目录 列表调用**api**导出excel文件

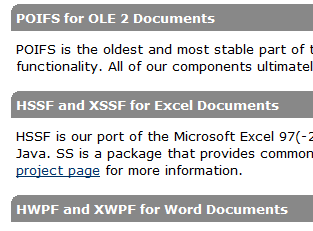
导出文件过程就是写文件过程

## Apache poi

使用3.10版本



Apache POI - the Java API for Microsoft Documents，Apache POI 是用Java编写的免费开源的跨平台的 Java API，它可以创建和维护操作各种符合Office Open XML（OOXML）标准和微软的OLE 2复合文档格式（OLE2）的Java API。用它可以使用Java读取和创建,修改MS Excel文件.而且,还可以使用Java读取和创建MS Word和MSPowerPoint文件。Apache POI 提供Java操作Excel解决方案（适用于Excel97-2008）。



学习目标：使用HSSF 和 XSSF操作excel文档。

## 使用HSSF 和 XSSF导出excel

Hssf导出excel

HSSF只操作.xls文件（97-03版本excel），一个sheet中行有限制，最大65536行。

第一步：创建workbook工作簿（excel文档）

Workbook wb = new HSSFWorkbook();

FileOutputStream fileOut = new FileOutputStream("workbook.xls");

wb.write(fileOut);

fileOut.close();

第二步：创建一个sheet工作表

Sheet sheet3 = wb.createSheet(safeName);

第三步：在sheet中创建row行

Row row = sheet.createRow((short)0);

第四步：在row行中创建cell单元格

Cell cell = row.createCell(0);

第五步：向cell中写数据

cell.setCellValue(1);

第六步：输出excel文件（写文件）

wb.write(fileOut);

fileOut.close();

测试代码：

// 创建文件输出流

FileOutputStream out = **new** FileOutputStream("d:/workbook.xls");

// 创建一个工作簿

Workbook wb = **new** HSSFWorkbook();

**for** (**int** j = 0; j < 1; j++) {

Sheet s = wb.createSheet();//创建1个sheet

wb.setSheetName(j, "sheet" + j);//指定sheet的名称

//xls文件最大支持65536行

**for** (**int** rownum = 0; rownum < 65536; rownum++) {//创建行,.xls一个sheet中的行数最大65535

// 创建一行

Row r = s.createRow(rownum);

**for** (**int** cellnum = 0; cellnum < 10; cellnum ++) {//一行创建10个单元格

// 在行里边创建单元格

Cell c = r.createCell(cellnum);

//向单元格写入数据

c.setCellValue(cellnum);

}

}

}

System.*out*.println("int..............");

wb.write(out);//输出文件内容

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

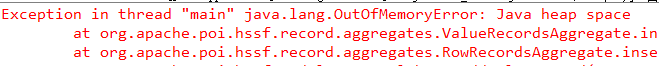
// **TODO** Auto-generated catch block

e.printStackTrace();

}

out.close();

使用上边的测试代码，在一个工作簿中导出多个sheet，出现内存溢出



出现内存溢出原因，HSSF工作原理，将excel中所有数据填充到java对象中，进行文件写操作。

缺点：

如果数据大，引起内存溢问题。

优点：

编程方便，如果数据量小，速度很快的。

XSSF导出 excel

XSSF操作03以上版本（07版本）excel，扩展名.xlsx，工作表行数没有限制

写excel文件步骤：

第一步：创建一个工作簿

SXSSFWorkbook wb = new SXSSFWorkbook(-1);

-1：关闭自动刷新

SXSSFWorkbook wb = new SXSSFWorkbook(XXXX);（自动刷新）

XXXX：保持内存中有XXXX条记录，超过部分写入磁盘

第二步：创建一个工作表

//创建一个sheet

Sheet sh = wb.createSheet();

第三步：在sheet中创建行

Row row = sh.createRow(rownum);

第四步：创建单元格

Cell cell = row.createCell(cellnum);

第五步：向单元格中写数据

cell.setCellValue(address);

第六步：将内容写入磁盘

由于-1设置关闭自动刷新

需要人工主动刷新

调用：

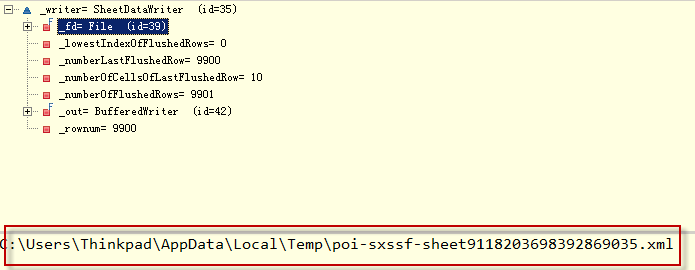
((SXSSFSheet)sh).flushRows(100);

第七步：输出文件

wb.write(out);//将临时写的文件合并，输出整个文件

跟踪代码：

向磁盘刷新数据，生成临时文件：



临时文件内容就是向excel中写的内容，

最后执行文件合并，将文件输出。

缺点：写数据时速度慢

优点：写大数据量时不会发生内存溢出

本系统采用XSSF导出药品目录 信息。

## 实现药品目录导出

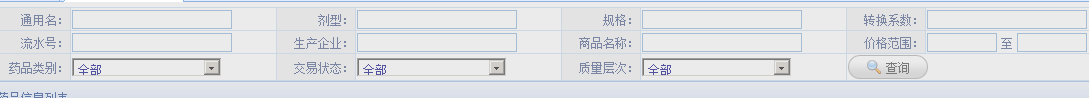
需求：

药品目录 导出功能，使用人：监督单位、医院、供货商

药品目录 添加/导入/删除：监督单位

操作步骤：

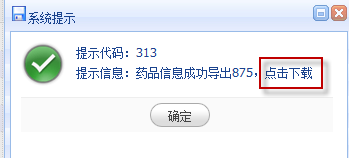
第一步：输入查询条件，查询出药品目录 信息



第二步：点击导出按钮

将符合查询条件的所有记录（不是一页的数据）全部导出。

第三步：下载导出文件



分析导出方法

分析一个技术，需要进行技术预研，确定使用XSSF导出excel文件。

确定采用自动刷新：

SXSSFWorkbook wb = new SXSSFWorkbook(XXXX);（自动刷新）

XXXX：保持内存中有XXXX条记录，超过部分写入磁盘

**对XSSF导出excel进行封装，写一个封装类，方便开发使用**。

ExcelExportSXXSSF.java

工具类的使用方法：

//导出文件存放的路径，并且是虚拟目录指向的路径

String filePath = "d:/upload/linshi/";

//导出文件的前缀

String filePrefix="ypxx";

//-1表示关闭自动刷新，手动控制写磁盘的时机，其它数据表示多少数据在内存保存，超过的则写入磁盘

**int** flushRows=100;

//定义导出数据的title

List<String> fieldNames=**new** ArrayList<String>();

fieldNames.add("流水号");

fieldNames.add("通用名");

fieldNames.add("价格");

//告诉导出类数据list中对象的属性，让ExcelExportSXXSSF通过反射获取对象的值

List<String> fieldCodes=**new** ArrayList<String>();

fieldCodes.add("bm");//药品流水号

fieldCodes.add("mc");//通用名

fieldCodes.add("price");//价格

//注意：fieldCodes和fieldNames个数必须相同且属性和title顺序一一对应，这样title和内容才一一对应

//开始导出，执行一些workbook及sheet等对象的初始创建

ExcelExportSXXSSF excelExportSXXSSF = ExcelExportSXXSSF.*start*(filePath, "/upload/", filePrefix, fieldNames, fieldCodes, flushRows);

//准备导出的数据，将数据存入list，且list中对象的字段名称必须是刚才传入ExcelExportSXXSSF的名称

List<Ypxx> list = **new** ArrayList<Ypxx>();

Ypxx ypxx1 = **new** Ypxx("001", "青霉素", 5);

Ypxx ypxx2 = **new** Ypxx("002", "感冒胶囊", 2.5f);

list.add(ypxx1);

list.add(ypxx2);

//执行导出

excelExportSXXSSF.writeDatasByObject(list);

//输出文件，返回下载文件的http地址

String webpath = excelExportSXXSSF.exportFile();

System.*out*.println(webpath);

药品目录导出实现

* Dao:

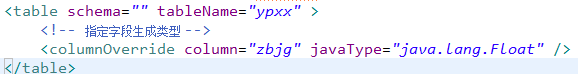
根据查询条件查询药品目录 信息，查询内容：



自定义mapper

逆向工程生成ypxx的mapper及po

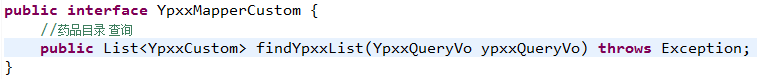
指定列生成 的java类型。



Sql：

主查询表：ypxx

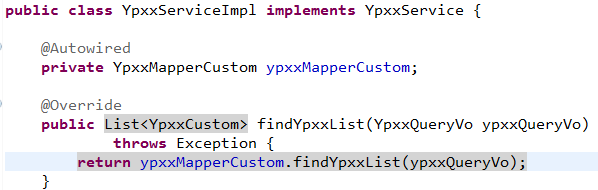
关联查询表：通过数据字典明细表查询交易状态对应的名称



* service

接口功能：药品目录 查询

接口参数：查询条件



* action

两个方法

导出页面展示；

导出功能提交：

调用封装类方法导出excel。

* 虚拟目录创建

虚拟目录：web请求路径

目标：将excel导出文件写到服务器的某个目录 ，用户通过客户端远程下载服务器上该目录 的内容。

服务器目录 ：

d:/upload/linshi/

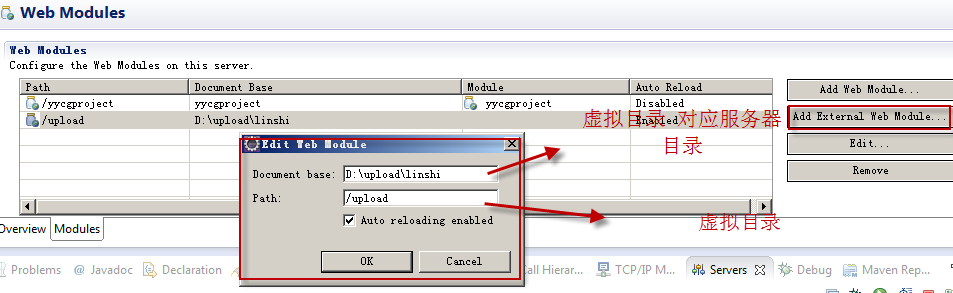
请求的路径就是虚拟目录http://ip:port/upload/

客户端

客户端从远程服务器上下载导出excel文件。

虚拟目录 创建：

通过图形界面创建：

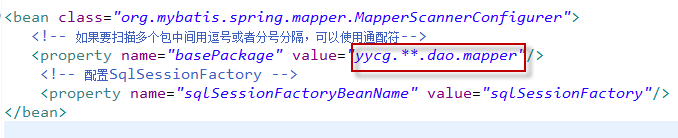


通过上边，实质操作tomcat下conf/server.xml



* 调试

Business下的mapper无法扫描到，修改applicationContext-base-dao.xml扫描器配置：



# 系统参数配置

系统参数是什么？

系统运行参数，一个系统运行所需要设置参数值。

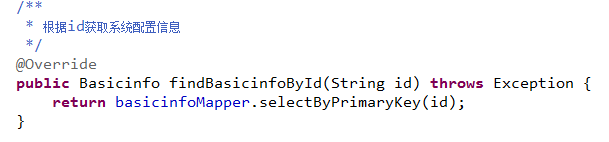
由管理员登陆系统后台进行设置，



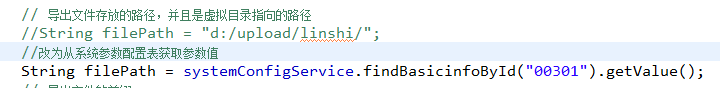
系统参数保存数据库BASICINFO

使用时，只需要根据BASICINFO表的id查询参数值。

在程序代码中，通过SystemConfigService，查询系统参数



使用方法：



# 药品目录导入

## 导入流程

将线下编辑好的excel导入系统中，从客户端导入服务端。

第一步：线下编辑导入文件

根据提前定义好的导入文件规则去定义，根据导入模块编写导入文件。

第二步：登陆系统，执行导入

将本地的文件上传到服务端

第三步：服务端程序解析导入文件内容

读文件过程。

## HSSF读excel文件

支持97-03版本的excel。

使用HSSF读文件步骤：

第一步： 创建一个workbook

//文件输入流

InputStream is = **new** FileInputStream("d:/test11.xls");

//创建hssf的workbook，将文件流传入workbook

HSSFWorkbook hssfWorkbook = **new** HSSFWorkbook(is);

第二步：读sheet

//得到workbook某个 的sheet，numSheet是sheet 的序号，序号从0开始

HSSFSheet hssfSheet = hssfWorkbook.getSheetAt(numSheet);

第三步：读sheet中的行

//读取每一行数据 ,rowNum指定行下标 从0开始

HSSFRow hssfRow = hssfSheet.getRow(rowNum);

第四步：读单元格的数据

//读取一行中某个单元格内容，cellNum指定单元格的下标，从0开始

HSSFCell cell = hssfRow.getCell(cellNum);

//调用单元格的get方法

cell.getStringCellValue()

通过读大数据文件，导致内存溢出



优点：读取少数据，读取速度很快

缺点：读取大数据量，读取速度很慢，可能导致内存溢出

内存溢出原因，将流中的数据全部加载到内存，进行输出。

上边的读方式是用户驱动模式，由用户指定想获取数据。

## HSSF事件驱动模式

事件驱动式，类似xml的sax解析。

需要实现HSSFListener接口。

原理：根据excel底层存储（07以版本采用xml存储，以下版本采用二进制）标签决定事件出发点。

目标：在解析完一行(row)数据时进行触发。

优点：读取大数据时，不会导致内存溢出

缺点：用户在解析数据时比较困难。读取数据时速度不快的，因为读取数据的同时根据每个标签进行事件触发。

## HSSF事件驱动读取文件封装类

底层抽象类

HxlsAbstract implements HSSFListener

//excel记录行操作方法，以sheet索引，行索引和行元素列表为参数，对sheet的一行元素进行操作，元素为String类型,rowlist存储了行数据

public abstract void optRows(int sheetIndex,int curRow, List<String> rowlist) throws Exception;

原理：在解析一行数据，在触发事件中设置rowlist(存储一行数据)

在行结束事件中，调用抽象方法（作用：解析一行数据）

工具类

HxlsRead extends HxlsAbstract

//数据导入接口

Private HxlsOptRowsInterface hxlsOptRowsInterface;

//抽象方法实现

public void optRows(int sheetIndex,int curRow, List<String> rowlist) throws Exception{

解析一行数据，调用hxlsOptRowsInterface(数据导入接口)

}

数据导入接口

HxlsOptRowsInterface

数据导入接口实现｛

实现药品目录 导入

接口接收到数据(解析一行数据)插入ypxx表

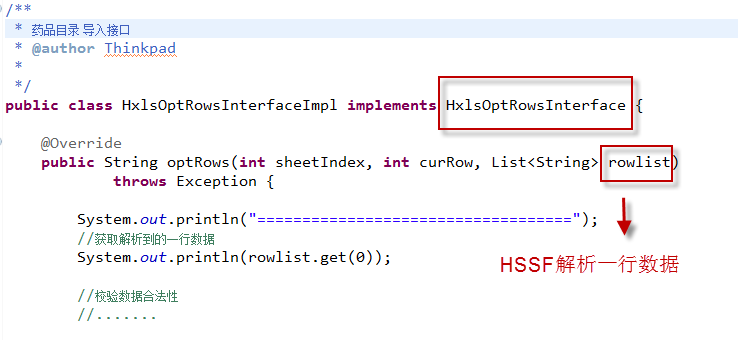
｝

注入

使用数据导入封装类方法：

第一步：实现HxlsOptRowsInterface（service）

根据具体业务需求，实现此接口



第二步：文件上传的action

第三步：在action调用工具类

使用HxlsRead工具类执行数据导入，需要上传文件