链滴

SSM 使用 POI 组件做 Excle 的批量导入导出

作者：kangaroo1122
原文链接：https://ld246.com/article/1556297071991
来源网站：链滴

批量导入使用的场景挺多的，自己毕设也涉及到了，做个记录。

导包在上一篇里已经提到了，这里就略过，重点记录一下实现的过程。

JSP界面

这个部分和上传word的一样，就是一个form

```
<form class="form-horizontal" id="fm_batchImport" method="post" enctype="multipart/fo
m-data">
    <input id="batchImport" name="enclosure" type="file"
        accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet,applicati
n/vnd.ms-excel">
    <a href="studentTemplate">模板下载</a>
    <button type="button" onClick="saveBatchImport()">提交</button>
</form>
```

还是和word一样，这里稍微做一个accept的验证，其实然并卵，后台还是需要判断。然后还是采用的jax上传，弄成formData。

Controller

两个方法，一个是作为导入，一个响应下载导入模板。

```
//批量导入学生信息
@RequestMapping(value="/batchImport",method=RequestMethod.POST)
@ResponseBody
public Integer batchImport(@ModelAttribute MultipartFile file, HttpServletRequest request, H
tpServletResponse response) throws Exception {
    int result = studentService.batchImportByAdmin(file, request, response);
    return result;
```

```
}

//模板下载
@RequestMapping("/studentTemplate")
@ResponseBody
public void studentTemplate(HttpServletRequest request, HttpServletResponse response) {
    // excel标题
    String[] title = { "学历名称", "专业名称", "学生姓名", "入学时间(yyyy-MM-dd)" };
    // excel文件名
    String fileName = "批量导入学生信息模板" + System.currentTimeMillis() + ".xls";
    // sheet名
    String sheetName = "学生模板";
    String[][] content = new String[0][title.length];
    //创建HSSFWorkbook
    HSSFWorkbook wb = ExcelUtil.getHSSFWorkbook(sheetName, title, content, null);
    // 响应到客户端
    try {
        this.setResponseHeader(response, fileName);
        OutputStream os = response.getOutputStream();
        wb.write(os);
        os.flush();
        os.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
private void setResponseHeader(HttpServletResponse response, String fileName) {
    try {
        try {
            fileName = new String(fileName.getBytes(), "ISO-8859-1");
        } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
        }
        response.setContentType("application/octet-stream;charset=ISO-8859-1");
        response.setHeader("Content-Disposition", "attachment;filename=" + fileName);
        response.addHeader("Pargam", "no-cache");
        response.addHeader("Cache-Control", "no-cache");
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

用到一个工具类ExcelUtil

```
package com.shms.util;
import java.io.IOException;
import java.io.InputStream;
import java.math.BigDecimal;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
```

```
import org.apache.poi.hssf.usermodel.HSSFDateUtil;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFCellStyle;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.ss.usermodel.HorizontalAlignment;

public class ExcelUtil {

    private final static String EXCEL2003L = ".xls"; // 2003- 版本的excel
    private final static String EXCEL2007U = ".xlsx"; // 2007+ 版本的excel

    /**
     * 导出Excel
     *
     * @param sheetName sheet名称
     * @param title     标题
     * @param values    内容
     * @param wb        HSSFWorkbook对象
     * @return
     */
    public static HSSFWorkbook getHSSFWorkbook(String sheetName, String[] title, String[][] v
lues, HSSFWorkbook wb) {
        // 第一步，创建一个HSSFWorkbook，对应一个Excel文件
        if (wb == null) {
            wb = new HSSFWorkbook();
        }
        // 第二步，在workbook中添加一个sheet,对应Excel文件中的sheet
        HSSFSheet sheet = wb.createSheet(sheetName);
        // 第三步，在sheet中添加表头第0行,注意老版本poi对Excel的行数列数有限制
        HSSFRow row = sheet.createRow(0);
        // 第四步，创建单元格，并设置表头 设置表头居中
        HSSFCellStyle style = wb.createCellStyle();
        style.setAlignment(HorizontalAlignment.CENTER); // 创建一个居中格式
        // 声明列对象
        HSSFCell cell = null;
        // 创建标题
        for (int i = 0; i < title.length; i++) {
            cell = row.createCell(i);
            cell.setCellValue(title[i]);
            cell.setCellStyle(style);
        }
        // 创建内容
        for (int i = 0; i < values.length; i++) {
            row = sheet.createRow(i + 1);
            for (int j = 0; j < values[i].length; j++) {
                // 将内容按顺序赋给对应的列对象
                row.createCell(j).setCellValue(values[i][j]);
```

```
            }
        }
        return wb;
    }

    /**
     * 导入excel 描述：获取IO流中的数据，组装成List<List<Object>>对象
     *
     * @param in,fileName
     * @return
     * @throws IOException
     */
    public List<List<Object>> getBankListByExcel(InputStream in, String fileName) throws Exc
ption {
        List<List<Object>> list = null;

        // 创建Excel工作薄
        Workbook work = this.getWorkbook(in, fileName);
        if (null == work) {
            throw new Exception("创建Excel工作薄为空！");
        }
        Sheet sheet = null; // Sheet页数
        Row row = null; // 行数
        Cell cell = null; // 列数

        list = new ArrayList<List<Object>>();
        // 遍历Excel中所有的sheet
        // 将最大的列数记录下来
        int lastCellNum = 0;
        for (int i = 0; i < work.getNumberOfSheets(); i++) {
            sheet = work.getSheetAt(i);
            if (sheet == null) {
                continue;
            }
            // 遍历当前sheet中的所有行
            for (int j = sheet.getFirstRowNum(); j <= sheet.getLastRowNum(); j++) {
                row = sheet.getRow(j);
                if (row == null || row.getFirstCellNum() == j) {
                    continue;
                }
                // 遍历所有的列
                List<Object> li = new ArrayList<Object>();
                // 比较当前行的列数跟表的最大的列数
                if (j == sheet.getFirstRowNum()) {
                    // 将第一行的列数设为最大
                    lastCellNum = row.getLastCellNum();
                } else {
                    lastCellNum = lastCellNum > row.getLastCellNum() ? lastCellNum : row.getLastCe
lNum();
                }
                for (int y = row.getFirstCellNum(); y < lastCellNum; y++) {
                    cell = row.getCell(y);
                    li.add(this.getValue(cell));
                }
```

```
            list.add(li);
        }
    }
    return list;
}

/**
 * 描述：根据文件后缀，自适应上传文件的版本
 *
 * @param inStr,fileName
 * @return
 * @throws Exception
 */
public Workbook getWorkbook(InputStream inStr, String fileName) throws Exception {
    Workbook wb = null;
    String fileType = fileName.substring(fileName.lastIndexOf("."));
    if (EXCEL2003L.equals(fileType)) {
        wb = new HSSFWorkbook(inStr); // 2003-
    } else if (EXCEL2007U.equals(fileType)) {
        wb = new XSSFWorkbook(inStr); // 2007+
    } else {
        throw new Exception("解析的文件格式有误！");
    }
    return wb;
}

/**
 * 描述：对表格中数值进行格式化
 *
 * @param cell
 * @return
 */
// 解决excel类型问题，获得数值
public String getValue(Cell cell) {
    String value = "";
    if (null == cell) {
        return value;
    }
    switch (cell.getCellType()) {
    // 数值型
    case NUMERIC:
        if (HSSFDateUtil.isCellDateFormatted(cell)) {
            // 如果是date类型则，获取该cell的date值
            Date date = HSSFDateUtil.getJavaDate(cell.getNumericCellValue());
            // 根据自己的实际情况，excel表中的时间格式是yyyy-MM-dd HH:mm:ss还是yyyy-MM-
d，或者其他类型
            SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
            // 由于方法的返回值类型为String，这里将Date类型转为String，便于统一返回数据
            value = format.format(date);
        } else {// 纯数字
            BigDecimal big = new BigDecimal(cell.getNumericCellValue());
            value = big.toString();
            // 解决1234.0 去掉后面的.0
            if (null != value && !"".equals(value.trim())) {
```

```
                String[] item = value.split("[.]");
                if (1 < item.length && "0".equals(item[1])) {
                    value = item[0];
                }
            }
        }
        break;
    // 字符串类型
    case STRING:
        value = cell.getStringCellValue().toString();
        break;
    // 公式类型
    case FORMULA:
        // 读公式计算值
        value = String.valueOf(cell.getNumericCellValue());
        if (value.equals("NaN")) {// 如果获取的数据值为非法值,则转换为获取字符串
            value = cell.getStringCellValue().toString();
        }
        break;
    // 布尔类型
    case BOOLEAN:
        value = " " + cell.getBooleanCellValue();
        break;
    default:
        value = cell.getStringCellValue().toString();
    }
    if ("null".endsWith(value.trim())) {
        value = "";
    }
    return value;
  }
}
```

在工具类中，有一点需要注意，cell.getCellType()的枚举书写，在4.x以前，是Cell.CELL_TYPE_STRING这种形式，在4.0.改成了CellType.STRING，在最新版中，变成了STRING。

其实模板下载，也可以作为后台数据导出Excel，只需要下载模板里边稍微改一下就行。比如导出成表

```
  String[][] content = new String[scoreList.size()][title.length]
  for (int i = 0; i < scoreList.size(); i++) {
      ScoreWrapper score = scoreList.get(i);
      content[i][0] = score.getcId();
      content[i][1] = score.getcName();
      content[i][2] = score.gettId();
      content[i][3] = score.gettName();
      content[i][4] = score.getsId();
      content[i][5] = score.getsName();
      content[i][6] = score.getAvgSubScore().toString();
  }
```

这样，也就把导出的功能完成了。

Service

具体的逻辑处理部分，读取表格数据。

```
@Override
public Integer batchImportByAdmin(MultipartFile file, HttpServletRequest request, HttpServle
Response response) throws Exception{
    if (file.isEmpty()) {
        try {
            throw new Exception("文件不存在！");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    //这里可以使用String originalFilename = file.getOriginalFilename();
    //originalFilename.endsWith(".xls") || originalFilename.endsWith(".xlsx")
    //做一个后缀名的判断，从而限制前端的文件格式
    InputStream in = null;
    try {
        in = file.getInputStream();
    } catch (IOException e) {
        e.printStackTrace();
    }

    List<List<Object>> listob = null;
    try {
        listob = new ExcelUtil().getBankListByExcel(in,file.getOriginalFilename());
    } catch (Exception e) {
        e.printStackTrace();
    }

    // 该处可调用mapper相应方法进行数据保存到数据库中
    for (int i = 0; i < listob.size(); i++) {
        List<Object> lo = listob.get(i);
        Student student = new Student();
        //这里可以做一些数据库查询操作等等..

        //这里的lo集合就是Excel表对应的列的数据（第一行默认是表头，不读取）
        String.valueOf(lo.get(0));//表示第i行第一列的数据，以此类推...

        //把数据塞入student对象，调用mapper方法进行存储
        student.setXXX();

        studentMapper.insertSelective(student);
    }
    return ...;
}
```

这样就完成了Excel的批量导入导出功能。