

* 学习目标

* 能够了解企业怎么操作Excel

- * 03:xls,07:xlsx

- * 数据存储格式，07：xlsx--核心是xml

- * poi

- * API：WorkBook,Sheet,Row,Cell,Font,CellStyle

* 能够掌握POI的基本操作

- * 创建excel文件，创建单元格，设置单元格样式，设置图片，查询excel表数据

* 能够使用POI导入Excel员工的信息

- * 文件上传，解析Excel，插入到数据

* 能够使用POI导出员工的信息

- * 从数据库中查询，创建ExcelAPI对象，通过流输出浏览器

* 能够掌握基于模板打印的POI报表导出

- * 有模板：在程序中获取样式，在创建新的单元格的时候，设置样式

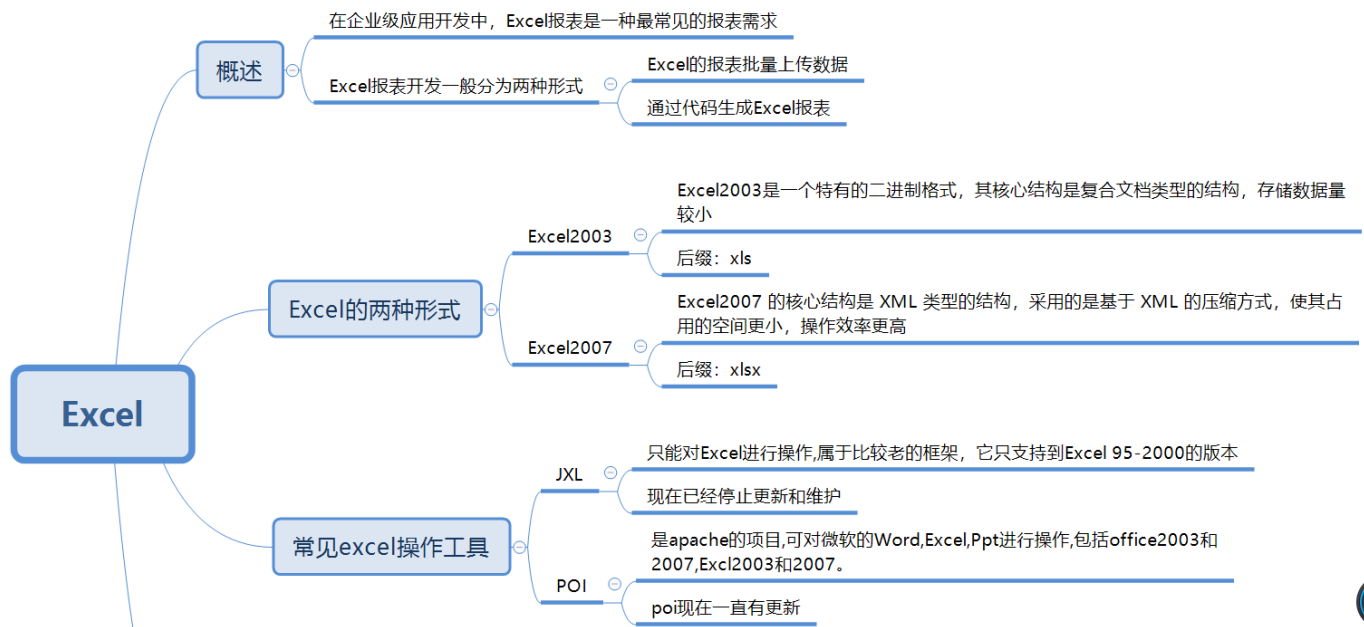
* 能够掌握自定义导入导出工具类

- * 提供开发效率，减少代码重复量，尽量避免Ctrl+C+V

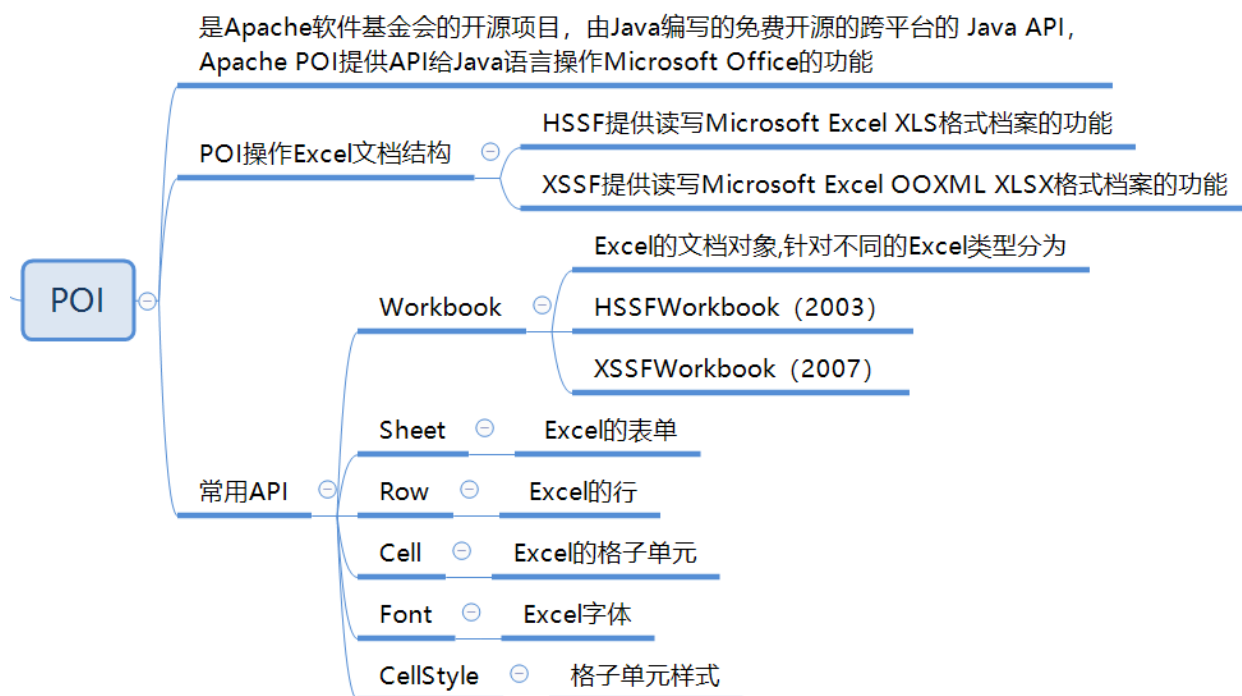
- * 前置知识：反射，自定义注解

* 回顾

* 能够了解企业怎么操作Excel



* 能够掌握POI的基本操作



```

1 * 添加依赖
2 <dependency>
3   <groupId>org.apache.poi</groupId>
4   <artifactId>poi</artifactId>
5   <version>4.0.1</version>
6 </dependency>
  
```

```
7 <dependency>
8   <groupId>org.apache.poi</groupId>
9   <artifactId>poi-ooxml</artifactId>
10  <version>4.0.1</version>
11 </dependency>
12 <dependency>
13   <groupId>org.apache.poi</groupId>
14   <artifactId>poi-ooxml-schemas</artifactId>
15   <version>4.0.1</version>
16 </dependency>
17 * 案例一（创建Excel）
18 @Test
19 public void test1() throws Exception {
20     //1.创建workbook工作簿
21     Workbook workbook=new XSSFWorkbook();
22     //2.创建表单Sheet
23     workbook.createSheet("test1");
24     //3.文件流
25     FileOutputStream fos=new FileOutputStream("D:\\wook666\\excel\\test.xlsx");
26     //4.写入文件
27     workbook.write(fos);
28     //5.释放资源
29     fos.close();
30 }
31
32 * 案例二（创建单元格）
33 @Test
34 public void test2() throws Exception {
35     Workbook workbook=new XSSFWorkbook();
36     Sheet sheet = workbook.createSheet("test1");
37     // 创建行对象，下标从0开始
38     Row row = sheet.createRow(2);
39     //创建单元格，从0开始
40     Cell cell = row.createCell(1);
41     //单元格设置内容
42     cell.setCellValue("亮哥教育");
43     FileOutputStream fos=new FileOutputStream("D:\\wook666\\excel\\test1.xlsx");
44     workbook.write(fos);
45     fos.close();
46 }
```

```
47
48 * 案例三(创建单元格, 样式)
49 @Test
50 public void test3() throws Exception {
51     Workbook workbook=new XSSFWorkbook();
52     Sheet sheet = workbook.createSheet("test1");
53     // 创建行对象, 下标从0开始
54     Row row = sheet.createRow(0);
55     //创建单元格, 从0开始
56     Cell cell = row.createCell(0);
57     // 设置边框
58     CellStyle cellStyle = workbook.createCellStyle();
59     cellStyle.setBorderBottom(BorderStyle.DOUBLE);
60     cellStyle.setBorderTop(BorderStyle.DOTTED);
61     cellStyle.setBorderLeft(BorderStyle.DASH_DOT);
62     cellStyle.setBorderRight(BorderStyle.HAIR);
63     // 设置字体
64     Font font = workbook.createFont();
65     font.setFontName("华文细黑");
66     font.setFontHeightInPoints((short) 32);
67     cellStyle.setFont(font);
68     cell.setCellStyle(cellStyle);
69     // 跨行跨列
70     CellRangeAddress region =new CellRangeAddress(0, 3, 0, 2);
71     sheet.addMergedRegion(region);
72     cellStyle.setAlignment(HorizontalAlignment.CENTER);//水平居中
73     cellStyle.setVerticalAlignment(VerticalAlignment.CENTER);//垂直居中
74     //单元格设置内容
75     cell.setCellValue("亮哥教育");
76     FileOutputStream fos=new FileOutputStream("D:\\wook666\\excel\\test2.xlsx");
77     workbook.write(fos);
78     fos.close();
79 }
80 * 案例四
81 //绘制图形
82 @Test
83 public void test4() throws Exception {
84     Workbook workbook=new XSSFWorkbook();
85     Sheet sheet = workbook.createSheet("test1");
86     FileInputStream fis=new FileInputStream("D:\\wook666\\excel\\logo.jpg");
```

```

87 //向Excel添加一张图片,并返回该图片在Excel中的图片集合中的下标
88 int pictureIndex = ((XSSFWorkbook) workbook).addPicture(fis, Workbook.PICTURE
89 //绘图工具类
90 CreationHelper creationHelper = workbook.getCreationHelper();
91 //创建一个绘图对象
92 Drawing<?> drawingPatriarch = sheet.createDrawingPatriarch();
93 //创建锚点, 设置图片坐标
94 ClientAnchor clientAnchor = creationHelper.createClientAnchor();
95 clientAnchor.setCol1(0);
96 clientAnchor.setRow1(0);
97 //创建图片
98 Picture picture = drawingPatriarch.createPicture(clientAnchor, pictureIndex);
99 picture.resize();
100 FileOutputStream fos=new FileOutputStream("D:\\wook666\\excel\\test3.xlsx");
101 workbook.write(fos);
102 fis.close();
103 fos.close();
104 }
105 * 案例五（读取Excel内容）
106 @Test
107 public void test5() throws Exception {
108 //创建工作簿
109 Workbook workbook=new XSSFWorkbook("D:\\wook666\\excel\\emps.xlsx");
110 // 获取sheet 从0开始
111 Sheet sheet = workbook.getSheetAt(0);
112 //获得最后一行
113 int lastRowNum = sheet.getLastRowNum();
114 for (int i = 0; i <=lastRowNum ; i++) {
115     Row row = sheet.getRow(i);
116     // 在一行中获得最后一列
117     short lastCellNum = row.getLastCellNum();
118     StringBuilder sb = new StringBuilder();
119     for (int j = 0; j < lastCellNum; j++) {
120         Cell cell = row.getCell(j);
121         Object value = ExcelUtils.getValue(cell);
122         sb.append(value).append(" ");
123     }
124     System.out.println(sb.toString());
125 }
126 }

```

```

127 public class ExcelUtils {
128     // 处理Excel数据类型
129     public static Object getValue(Cell cell) {
130         Object value = null;
131         switch (cell.getCellType()) {
132             case STRING:
133                 value = cell.getStringCellValue();
134                 break;
135             case BOOLEAN:
136                 value = cell.getBooleanCellValue();
137                 break;
138             case NUMERIC://数字类型（包含日期和普通数字）
139                 if (DateUtil.isCellDateFormatted(cell)) {
140                     value = cell.getDateCellValue();
141                 } else {
142                     value = cell.getNumericCellValue();
143                 }
144                 break;
145             case FORMULA:
146                 value = cell.getCellFormula();
147                 break;
148             default:
149                 break;
150         }
151         return value;
152     }
153
154 }

```

* 能够使用POI导入Excel员工的信息

```

1 * 配置
2 @Bean("multipartResolver")
3 public CommonsMultipartResolver multipartResolver(){
4     CommonsMultipartResolver commonsMultipartResolver=new CommonsMultipartResolv
5     //50M
6     commonsMultipartResolver.setMaxUploadSize(52428800);

```

```
7     return commonsMultipartResolver;
8 }
9 * 实体
10 @Data
11 @AllArgsConstructor
12 @NoArgsConstructor
13 public class Employee {
14     private int id;
15     private String empno;
16     private String name;
17     private String sex;
18     private int age;
19     private String job;
20     private int departid;
21     private List<Goods> goods;
22 }
23
24 * Dao
25 public interface EmployeeDao {
26     @Insert("INSERT INTO employee(empno,NAME,sex,age,job,departid)
27         VALUES(#{empno},#{name},#{sex},#{age},#{job},#{departid})")
28     void save(Employee employee);
29 }
30 public interface DepartmentDao {
31     @Update("UPDATE department SET dnum=dnum+1 WHERE deptno=#{ndid}")
32     void updateDepartmentByIncrease(Map<String, Object> params);
33 }
34 * Service
35 public interface ExcelService {
36     void importExcel(List<Employee> employees);
37 }
38 @Service
39 public class ExcelServiceImpl implements ExcelService {
40     @Autowired
41     private EmployeeDao employeeDao;
42     @Autowired
43     private DepartmentDao departmentDao;
44     @Transactional
45     @Override
46     public void importExcel(List<Employee> employees) {
```

```
47     for (Employee employee : employees) {
48         employeeDao.save(employee);
49         Map<String,Object> params=new HashMap<>();
50         params.put("ndid",employee.getDepartid());
51         departmentDao.updateDepartmentByIncrease(params);
52     }
53 }
54 }
55 * Controller
56 @Controller
57 @RequestMapping("/emp")
58 public class EmployeeController {
59     @RequestMapping("/uploadExcel")
60     public String fileUpload(MultipartFile uploadFile, Model model){
61         try {
62             //创建workbook工作簿
63             Workbook workbook=new XSSFWorkbook(uploadFile.getInputStream());
64             // 获取sheet 从0开始
65             Sheet sheet = workbook.getSheetAt(0);
66             //获得最后一行
67             int lastRowNum = sheet.getLastRowNum();
68             List<Employee> employees=new ArrayList<>();
69             for (int i = 1; i <=lastRowNum ; i++) {
70                 Row row = sheet.getRow(i);
71                 // 在一行中获得最后一列
72                 short lastCellNum = row.getLastCellNum();
73                 Object[] objs=new Object[lastCellNum];
74                 for (int j = 0; j < lastCellNum; j++) {
75                     Cell cell = row.getCell(j);
76                     Object value = ExcelUtils.getValue(cell);
77                     objs[j]=value;
78                 }
79                 Employee employee=new Employee();
80                 employee.setEmpno((String) objs[0]);
81                 employee.setName((String) objs[1]);
82                 employee.setSex((String) objs[2]);
83                 employee.setAge(((Double) objs[3]).intValue());
84                 employee.setJob((String) objs[4]);
85                 employee.setDepartid(((Double) objs[5]).intValue());
86                 employees.add(employee);
```



```

87     }
88     excelService.importExcel(employees);
89     model.addAttribute("result","上传成功");
90 }catch (Exception e){
91     e.printStackTrace();
92     model.addAttribute("result","上传失败");
93 }
94     return "uploadsucces";
95 }
96 }
97
98 * 页面
99 * 在webapp下
100 <%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="fa
101 <html>
102 <head>
103     <title>测试</title>
104 </head>
105 <body>
106 <form action="${pageContext.request.contextPath}/emp/uploadExcel" method="post"
107     enctype="multipart/form-data">
108     图片: <input type="file" name="uploadFile"/><br/>
109     <input type="submit" value="上传"/>
110 </form>
111 </body>
112 </html>
113 * 在/WEB-INF/jsp/下
114 <%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="fa
115 <html>
116 <body>
117 <h2>文件${result}</h2>
118 </body>
119 </html>

```

* 能够使用POI导出员工的信息

```

1 * dao
2 public interface EmployeeDao {
3     @Select("SELECT empno,NAME,sex,age,job,departid FROM employee")

```

```
4     List<Employee> selectEmps();
5 }
6
7 * service
8 public interface ExcelService {
9     List<Employee> exportExcel();
10 }
11 * Controller
12 @Controller
13 @RequestMapping("/emp")
14 public class EmployeeController {
15     @Autowired
16     private ExcelService excelService;
17     @RequestMapping("/exportExcel")
18     public void exportExcel(HttpServletResponse response, Model model){
19         try {
20             List<Employee> employees = excelService.exportExcel();
21             Workbook workbook = new XSSFWorkbook();
22             Sheet sheet = workbook.createSheet("一月");
23             String[] titles = {"员工编号", "姓名", "性别", "年龄", "职位", "部门编
24             Row row = sheet.createRow(0);
25             for (int i = 0; i < titles.length; i++) {
26                 Cell cell = row.createCell(i);
27                 cell.setCellValue(titles[i]);
28             }
29             for (int i = 0; i < employees.size(); i++) {
30                 row = sheet.createRow(i + 1);
31                 Employee employee = employees.get(i);
32                 Cell empnoCell = row.createCell(0);
33                 empnoCell.setCellValue(employee.getEmpno());
34                 Cell nameCell = row.createCell(1);
35                 nameCell.setCellValue(employee.getName());
36                 Cell sexCell = row.createCell(2);
37                 sexCell.setCellValue(employee.getSex());
38                 Cell ageCell = row.createCell(3);
39                 ageCell.setCellValue(employee.getAge());
40                 Cell jobCell = row.createCell(4);
41                 jobCell.setCellValue(employee.getJob());
42                 Cell departIdCell = row.createCell(5);
43                 departIdCell.setCellValue(employee.getDepartid());
```

```

44         }
45         String fileName = URLEncoder.encode("员工表.xlsx", "UTF-8");
46         response.setContentType("application/octet-stream");
47         response.setHeader("content-disposition", "attachment;filename=" +
48         response.setHeader("filename", fileName);
49         workbook.write(response.getOutputStream());
50     }catch (Exception e){
51         e.printStackTrace();
52     }
53 }
54 }
55
56 * 访问测试: http://localhost:8080/ssm02/emp/exportExcel

```

* 能够掌握基于模板打印的POI报表导出

```

1  * 在企业开发中，我们可以先做模板，在程序获取模板的样式
2  生成模板格式的excel文件
3  * 代码
4  @RequestMapping("/exportExcelStyle")
5  public void exportExcelStyle(HttpServletResponse response){
6      try {
7          List<Employee> employees = excelService.exportExcel();
8          Workbook workbook = new XSSFWorkbook(
9              Objects.requireNonNull(EmployeeController.class.
10                 getClassLoader().getResourceAsStream("/excel/template.xlsx")
11             Sheet sheet = workbook.getSheetAt(0);
12             Row row1= sheet.getRow(2);
13             //获取模板的样式
14             CellStyle[] cellStyles=new CellStyle[row1.getLastCellNum()];
15             for (int i = 0; i < cellStyles.length ; i++) {
16                 cellStyles[i]=row1.getCell(i).getCellStyle();
17             }
18             for (int i = 0; i < employees.size(); i++) {
19                 Row row = sheet.createRow(i + 2);
20                 Employee employee = employees.get(i);
21                 Cell empnoCell = row.createCell(0);
22                 empnoCell.setCellValue(employee.getEmpno());
23                 empnoCell.setCellStyle(cellStyles[0]);

```

```

24         Cell nameCell = row.createCell(1);
25         nameCell.setCellValue(employee.getName());
26         nameCell.setCellStyle(cellStyles[1]);
27         Cell sexCell = row.createCell(2);
28         sexCell.setCellValue(employee.getSex());
29         sexCell.setCellStyle(cellStyles[2]);
30         Cell ageCell = row.createCell(3);
31         ageCell.setCellValue(employee.getAge());
32         ageCell.setCellStyle(cellStyles[3]);
33         Cell jobCell = row.createCell(4);
34         jobCell.setCellValue(employee.getJob());
35         jobCell.setCellStyle(cellStyles[4]);
36         Cell departIdCell = row.createCell(5);
37         departIdCell.setCellValue(employee.getDepartid());
38         departIdCell.setCellStyle(cellStyles[5]);
39     }
40     String fileName = URLEncoder.encode("员工表2.xlsx", "UTF-8");
41     response.setContentType("application/octet-stream");
42     response.setHeader("content-disposition", "attachment;filename=" +
43     response.setHeader("filename", fileName);
44     workbook.write(response.getOutputStream());
45 }catch (Exception e){
46     e.printStackTrace();
47 }
48 }

```

* 能够掌握自定义导入导出工具类

```

1  * 案例一：（导出工具类封装）
2  * 自定义注解
3  @Retention(RetentionPolicy.RUNTIME)
4  @Target(ElementType.FIELD)
5  public @interface ExcelAttribute {
6      /** 列序号 */
7      int sort();
8  }
9
10 * 导出工具类封装
11 public class ExcelExportUtil<T> {

```

```

12     public void export(HttpServletResponse response,
13         Map<String,Object> params) throws Exception{
14         String classFilePath= (String) params.get("classFilePath");
15         Integer styleIndex= (Integer) params.get("styleIndex");
16         Integer rowIndex= (Integer) params.get("rowIndex");
17         List<T> objs= (List<T>) params.get("objs");
18         String fileName= (String) params.get("fileName");
19         Workbook workbook = new XSSFWorkbook(Objects.requireNonNull(ExcelExport
20             getClassLoader().getResourceAsStream(classFilePath)));
21         Sheet sheet = workbook.getSheetAt(0);
22         Row row1= sheet.getRow(styleIndex);
23         CellStyle[] cellStyles=new CellStyle[row1.getLastCellNum()];
24         for (int i = 0; i < cellStyles.length ; i++) {
25             cellStyles[i]=row1.getCell(i).getCellStyle();
26         }
27         for (int i = 0; i < objs.size(); i++) {
28             Row row = sheet.createRow(i + rowIndex);
29             T t = objs.get(i);
30             for (int j = 0; j < cellStyles.length ; j++) {
31                 Cell cell = row.createCell(j);
32                 cell.setCellStyle(cellStyles[j]);
33                 Class<?> clazz = t.getClass();
34                 Field[] fields = clazz.getDeclaredFields();
35                 for (Field field : fields) {
36                     if(field.isAnnotationPresent(ExcelAttribute.class)){
37                         field.setAccessible(true);
38                         ExcelAttribute excelAttribute =
39                             field.getAnnotation(ExcelAttribute.class);
40                         if(j==excelAttribute.sort()){
41                             cell.setCellValue(field.get(t).toString());
42                         }
43                     }
44                 }
45             }
46         }
47         fileName = URLEncoder.encode(fileName, "UTF-8");
48         response.setContentType("application/octet-stream");
49         response.setHeader("content-disposition", "attachment;filename=" + file
50         response.setHeader("filename", fileName);
51         workbook.write(response.getOutputStream());

```

```

52     }
53 }
54
55 * Controller
56 @RequestMapping("/exportExcelStyle2")
57 public void exportExcelStyle2(HttpServletResponse response) throws Exception {
58     List<Employee> employees = excelService.exportExcel();
59     ExcelExportUtil excelExportUtil=new ExcelExportUtil();
60     Map<String,Object> params=new HashMap<String,Object>();
61     params.put("classFilePath","/excel/template.xlsx");
62     params.put("styleIndex",2);
63     params.put("rowIndex",2);
64     params.put("objs",employees);
65     params.put("fileName","员工表3.xlsx");
66     excelExportUtil.export(response,params);
67 }
68
69 * 案例二：（导入工具类封装）
70 public class ExcelImportUtil<T> {
71     public List<T> excelImportUtil(InputStream is,Class<T> clazz,int rowIndex)
72     throws Exception {
73         //创建workbook工作簿
74         Workbook workbook = new XSSFWorkbook(is);
75         // 获取sheet 从0开始
76         Sheet sheet = workbook.getSheetAt(0);
77         //获得最后一行
78         int lastRowNum = sheet.getLastRowNum();
79         List<T> list = new ArrayList<>();
80         for (int i = rowIndex; i <= lastRowNum; i++) {
81             Row row = sheet.getRow(i);
82             // 在一行中获得最后一列
83             short lastCellNum = row.getLastCellNum();
84             Object[] objs = new Object[lastCellNum];
85             T t = clazz.newInstance();
86             Field[] fields = clazz.getDeclaredFields();
87             for (int j = 0; j < lastCellNum; j++) {
88                 Cell cell = row.getCell(j);
89                 Object value = ExcelUtils.getValue(cell);
90                 objs[j] = value;
91             }

```

```

92         for (int j = 0; j < lastCellNum; j++) {
93             for (Field field : fields) {
94                 field.setAccessible(true);
95                 if(field.isAnnotationPresent(ExcelAttribute.class)){
96                     ExcelAttribute excelAttribute =
97                         field.getAnnotation(ExcelAttribute.class);
98                     int sort = excelAttribute.sort();
99                     if(j==sort){
100                         // todo:类型, 根据需要添加
101                         if(objs[j] instanceof Double){
102                             // employee.setAge(((Double) objs[3]).intValue());
103                             field.set(t,((Double) objs[j]).intValue());
104                         }else{
105                             field.set(t,objs[j]);
106                         }
107                     }
108                 }
109             }
110         }
111     }
112     list.add(t);
113 }
114 return list;
115 }
116 }
117 * Controller
118 @RequestMapping("/uploadExcel2")
119 public String fileUpload2(MultipartFile uploadFile, Model model){
120     try {
121         ExcelImportUtil<Employee> excelImportUtil=new ExcelImportUtil<>();
122         List<Employee> employees = excelImportUtil.
123             excelImportUtil(uploadFile.getInputStream(), Employee.class, 1);
124         excelService.importExcel(employees);
125         model.addAttribute("result","上传成功");
126     }catch (Exception e){
127         e.printStackTrace();
128         model.addAttribute("result","上传失败");
129     }
130     return "uploadsucces";
131 }

```

