* 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

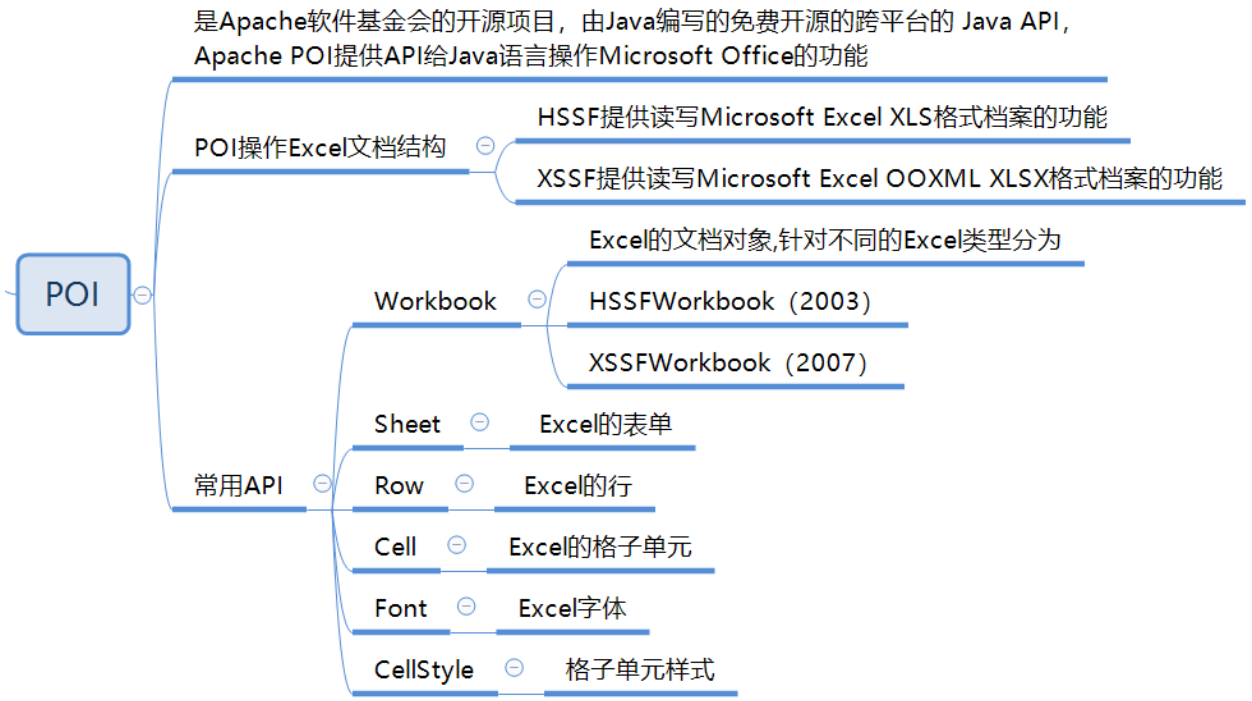
     * 前置知识：反射，自定义注解

---

 * 回顾

 * 能够了解企业怎么操作Excel

* 能够掌握POI的基本操作




```
* 添加依赖
<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi</artifactId>
  <version>4.0.1</version>
</dependency>
```

```
<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi-ooxml</artifactId>
  <version>4.0.1</version>
</dependency>
<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi-ooxml-schemas</artifactId>
  <version>4.0.1</version>
</dependency>
* 案例一（创建Excel）
 @Test
public void test1() throws Exception {
  //1.创建workbook工作簿
  Workbook workbook=new XSSFWorkbook();
  //2.创建表单Sheet
  workbook.createSheet("test1");
  //3.文件流
  FileOutputStream fos=new FileOutputStream("D:\\wook666\\excel\\test.xlsx");
  //4.写入文件
  workbook.write(fos);
  //5.释放资源
  fos.close();
}

* 案例二（创建单元格）
@Test
public void test2() throws Exception {
  Workbook workbook=new XSSFWorkbook();
  Sheet sheet = workbook.createSheet("test1");
  // 创建行对象，下标从0开始
  Row row = sheet.createRow(2);
  //创建单元格，从0开始
  Cell cell = row.createCell(1);
  //单元格设置内容
  cell.setCellValue("亮哥教育");
  FileOutputStream fos=new FileOutputStream("D:\\wook666\\excel\\test1.xlsx");
  workbook.write(fos);
  fos.close();
}
```

```
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");
```

```
  //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
  int pictureIndex = ((XSSFWorkbook) workbook).addPicture(fis, Workbook.PICTURE
  //绘图工具类
  CreationHelper creationHelper = workbook.getCreationHelper();
  //创建一个绘图对象
  Drawing<?> drawingPatriarch = sheet.createDrawingPatriarch();
  //创建锚点，设置图片坐标
  ClientAnchor clientAnchor = creationHelper.createClientAnchor();
  clientAnchor.setCol1(0);
  clientAnchor.setRow1(0);
  //创建图片
  Picture picture = drawingPatriarch.createPicture(clientAnchor, pictureIndex);
  picture.resize();
  FileOutputStream fos=new FileOutputStream("D:\\wook666\\excel\\test3.xlsx");
  workbook.write(fos);
  fis.close();
  fos.close();
}
* 案例五（读取Excel内容）
@Test
public void test5() throws Exception {
 //创建workbook工作簿
 Workbook workbook=new XSSFWorkbook("D:\\wook666\\excel\\emps.xlsx");
 // 获取sheet 从0开始
 Sheet sheet = workbook.getSheetAt(0);
 //获得最后一行
 int lastRowNum = sheet.getLastRowNum();
 for (int i = 0; i <=lastRowNum ; i++) {
       Row row = sheet.getRow(i);
       // 在一行中获得最后一列
       short lastCellNum = row.getLastCellNum();
       StringBuilder sb = new StringBuilder();
       for (int j = 0; j < lastCellNum; j++) {
           Cell cell = row.getCell(j);
           Object value = ExcelUtils.getValue(cell);
           sb.append(value).append("   ");
         }
          System.out.println(sb.toString());
        }
}
```

```java
public class ExcelUtils {
    // 处理Excel数据类型
    public static Object getValue(Cell cell) {
        Object value = null;
        switch (cell.getCellType()) {
            case STRING:
                value = cell.getStringCellValue();
                break;
            case BOOLEAN:
                value = cell.getBooleanCellValue();
                break;
            case NUMERIC://数字类型（包含日期和普通数字）
                if (DateUtil.isCellDateFormatted(cell)) {
                    value = cell.getDateCellValue();
                } else {
                    value = cell.getNumericCellValue();
                }
                break;
            case FORMULA:
                value = cell.getCellFormula();
                break;
            default:
                break;
        }
        return value;
    }

}
```

* 能够使用POI导入Excel员工的信息

```java
* 配置
@Bean("multipartResolver")
public CommonsMultipartResolver multipartResolver(){
   CommonsMultipartResolver commonsMultipartResolver=new CommonsMultipartResolv
   //50M
   commonsMultipartResolver.setMaxUploadSize(52428800);
```

```
   return commonsMultipartResolver;
}
* 实体
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Employee {
    private int id;
    private String empno;
    private String name;
    private String sex;
    private int age;
    private String job;
    private int departid;
    private List<Goods> goods;
}

* Dao
public interface EmployeeDao {
    @Insert("INSERT INTO employee(empno,NAME,sex,age,job,departid)
            VALUES(#{empno},#{name},#{sex},#{age},#{job},#{departid})")
    void save(Employee employee);
}
public interface DepartmentDao {
    @Update("UPDATE department SET dnum=dnum+1 WHERE deptno=#{ndid}")
    void updateDepartmentByIncrease(Map<String, Object> params);
}
* Service
public interface ExcelService {
     void importExcel(List<Employee> employees);
}
@Service
public class ExcelServiceImpl implements ExcelService {
    @Autowired
    private EmployeeDao employeeDao;
    @Autowired
    private DepartmentDao departmentDao;
    @Transactional
    @Override
    public void importExcel(List<Employee> employees) {
```

```
        for (Employee employee : employees) {
            employeeDao.save(employee);
            Map<String,Object> params=new HashMap<>();
            params.put("ndid",employee.getDepartid());
            departmentDao.updateDepartmentByIncrease(params);
        }
    }
}
* Controller
@Controller
@RequestMapping("/emp")
public class EmployeeController {
    @RequestMapping("/uploadExcel")
    public String fileUpload(MultipartFile uploadFile, Model model){
        try {
        //创建workbook工作簿
        Workbook workbook=new XSSFWorkbook(uploadFile.getInputStream());
        // 获取sheet 从0开始
        Sheet sheet = workbook.getSheetAt(0);
        //获得最后一行
        int lastRowNum = sheet.getLastRowNum();
        List<Employee> employees=new ArrayList<>();
        for (int i = 1; i <=lastRowNum ; i++) {
            Row row = sheet.getRow(i);
            // 在一行中获得最后一列
            short lastCellNum = row.getLastCellNum();
            Object[] objs=new Object[lastCellNum];
            for (int j = 0; j < lastCellNum; j++) {
                Cell cell = row.getCell(j);
                Object value = ExcelUtils.getValue(cell);
                objs[j]=value;
            }
            Employee employee=new Employee();
            employee.setEmpno((String) objs[0]);
            employee.setName((String) objs[1]);
            employee.setSex((String) objs[2]);
            employee.setAge(((Double) objs[3]).intValue());
            employee.setJob((String) objs[4]);
            employee.setDepartid(((Double) objs[5]).intValue());
            employees.add(employee);
```

```
        }
            excelService.importExcel(employees);
            model.addAttribute("result","上传成功");
        }catch (Exception e){
            e.printStackTrace();
            model.addAttribute("result","上传失败");
        }
        return "uploadsuccess";
    }
}

* 页面
* 在webapp下
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="fal
<html>
<head>
    <title>测试</title>
</head>
<body>
<form action="${pageContext.request.contextPath}/emp/uploadExcel" method="post"
      enctype="multipart/form-data">
    图片：<input type="file" name="uploadFile"/><br/>
    <input type="submit" value="上传"/>
</form>
</body>
</html>
* 在/WEB-INF/jsps/下
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="fal
<html>
<body>
<h2>文件${result}</h2>
</body>
</html>
```

* 能够使用POI导出员工的信息

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

```
    List<Employee> selectEmps();
}

* service
public interface ExcelService {
     List<Employee> exportExcel();
}
* Controller
@Controller
@RequestMapping("/emp")
public class EmployeeController {
    @Autowired
    private ExcelService excelService;
    @RequestMapping("/exportExcel")
    public void exportExcel(HttpServletResponse response,Model model){
        try {
            List<Employee> employees = excelService.exportExcel();
            Workbook workbook = new XSSFWorkbook();
            Sheet sheet = workbook.createSheet("一月");
            String[] titles = {"员工编号", "姓名", "性别", "年龄", "职位", "部门编
            Row row = sheet.createRow(0);
            for (int i = 0; i < titles.length; i++) {
                Cell cell = row.createCell(i);
                cell.setCellValue(titles[i]);
            }
            for (int i = 0; i < employees.size(); i++) {
                row = sheet.createRow(i + 1);
                Employee employee = employees.get(i);
                Cell empnoCell = row.createCell(0);
                empnoCell.setCellValue(employee.getEmpno());
                Cell nameCell = row.createCell(1);
                nameCell.setCellValue(employee.getName());
                Cell sexCell = row.createCell(2);
                sexCell.setCellValue(employee.getSex());
                Cell ageCell = row.createCell(3);
                ageCell.setCellValue(employee.getAge());
                Cell jobCell = row.createCell(4);
                jobCell.setCellValue(employee.getJob());
                Cell departIdCell = row.createCell(5);
                departIdCell.setCellValue(employee.getDepartid());
```

```
            }
            String fileName = URLEncoder.encode("员工表.xlsx", "UTF-8");
            response.setContentType("application/octet-stream");
            response.setHeader("content-disposition", "attachment;filename=" +
            response.setHeader("filename", fileName);
            workbook.write(response.getOutputStream());
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}

* 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

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

```
                Cell nameCell = row.createCell(1);
                nameCell.setCellValue(employee.getName());
                nameCell.setCellStyle(cellStyles[1]);
                Cell sexCell = row.createCell(2);
                sexCell.setCellValue(employee.getSex());
                sexCell.setCellStyle(cellStyles[2]);
                Cell ageCell = row.createCell(3);
                ageCell.setCellValue(employee.getAge());
                ageCell.setCellStyle(cellStyles[3]);
                Cell jobCell = row.createCell(4);
                jobCell.setCellValue(employee.getJob());
                jobCell.setCellStyle(cellStyles[4]);
                Cell departIdCell = row.createCell(5);
                departIdCell.setCellValue(employee.getDepartid());
                departIdCell.setCellStyle(cellStyles[5]);
            }
            String fileName = URLEncoder.encode("员工表2.xlsx", "UTF-8");
            response.setContentType("application/octet-stream");
            response.setHeader("content-disposition", "attachment;filename=" +
            response.setHeader("filename", fileName);
            workbook.write(response.getOutputStream());
        }catch (Exception e){
            e.printStackTrace();
        }
    }
```

* 能够掌握自定义导入导出工具类

```
* 案例一：（导出工具类封装）
* 自定义注解
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
public @interface ExcelAttribute {
    /** 列序号 */
    int sort();
}

* 导出工具类封装
public class ExcelExportUtil<T> {
```

```
    public void export(HttpServletResponse response,
        Map<String,Object> params) throws Exception{
        String classFilePath= (String) params.get("classFilePath");
        Integer styleIndex= (Integer) params.get("styleIndex");
        Integer rowIndex= (Integer) params.get("rowIndex");
        List<T> objs= (List<T>) params.get("objs");
        String fileName= (String) params.get("fileName");
        Workbook workbook = new XSSFWorkbook(Objects.requireNonNull(ExcelExport
                getClassLoader().getResourceAsStream(classFilePath)));
        Sheet sheet = workbook.getSheetAt(0);
        Row row1= sheet.getRow(styleIndex);
        CellStyle[] cellStyles=new CellStyle[row1.getLastCellNum()];
        for (int i = 0; i <cellStyles.length ; i++) {
            cellStyles[i]=row1.getCell(i).getCellStyle();
        }
        for (int i = 0; i < objs.size(); i++) {
            Row row = sheet.createRow(i + rowIndex);
            T t = objs.get(i);
            for (int j = 0; j <cellStyles.length ; j++) {
                Cell cell = row.createCell(j);
                cell.setCellStyle(cellStyles[j]);
                Class<?> clazz = t.getClass();
                Field[] fields = clazz.getDeclaredFields();
                for (Field field : fields) {
                    if(field.isAnnotationPresent(ExcelAttribute.class)){
                        field.setAccessible(true);
                        ExcelAttribute excelAttribute =
                            field.getAnnotation(ExcelAttribute.class);
                        if(j==excelAttribute.sort()){
                            cell.setCellValue(field.get(t).toString());
                        }
                    }
                }
            }
        }
        fileName = URLEncoder.encode(fileName, "UTF-8");
        response.setContentType("application/octet-stream");
        response.setHeader("content-disposition", "attachment;filename=" + file
        response.setHeader("filename", fileName);
        workbook.write(response.getOutputStream());
```

```
    }
}

* Controller
 @RequestMapping("/exportExcelStyle2")
public void exportExcelStyle2(HttpServletResponse response) throws Exception {
      List<Employee> employees = excelService.exportExcel();
      ExcelExportUtil excelExportUtil=new ExcelExportUtil();
      Map<String,Object> params=new HashMap<String,Object>();
      params.put("classFilePath","/excel/template.xlsx");
      params.put("styleIndex",2);
      params.put("rowIndex",2);
      params.put("objs",employees);
      params.put("fileName","员工表3.xlsx");
      excelExportUtil.export(response,params);
}

* 案例二：（导入工具类封装）
public class ExcelImportUtil<T> {
    public List<T> excelImportUtil(InputStream is,Class<T> clazz,int rowIndex)
    throws Exception {
        //创建workbook工作簿
        Workbook workbook = new XSSFWorkbook(is);
        // 获取sheet 从0开始
        Sheet sheet = workbook.getSheetAt(0);
        //获得最后一行
        int lastRowNum = sheet.getLastRowNum();
        List<T> list = new ArrayList<>();
        for (int i = rowIndex; i <= lastRowNum; i++) {
            Row row = sheet.getRow(i);
            // 在一行中获得最后一列
            short lastCellNum = row.getLastCellNum();
            Object[] objs = new Object[lastCellNum];
            T t = clazz.newInstance();
            Field[] fields = clazz.getDeclaredFields();
            for (int j = 0; j < lastCellNum; j++) {
                Cell cell = row.getCell(j);
                Object value = ExcelUtils.getValue(cell);
                objs[j] = value;
            }
```

```
            for (int j = 0; j < lastCellNum; j++) {
                for (Field field : fields) {
                    field.setAccessible(true);
                    if(field.isAnnotationPresent(ExcelAttribute.class)){
                        ExcelAttribute excelAttribute =
                    field.getAnnotation(ExcelAttribute.class);
                        int sort = excelAttribute.sort();
                        if(j==sort){
                            // todo:类型，根据需要添加
                            if(objs[j] instanceof Double){
                             // employee.setAge(((Double) objs[3]).intValue());
                                field.set(t,((Double) objs[j]).intValue());
                            }else{
                                field.set(t,objs[j]);
                            }

                        }
                    }
                }
            }
            list.add(t);
        }
        return list;
    }
}
* Controller
 @RequestMapping("/uploadExcel2")
public String fileUpload2(MultipartFile uploadFile, Model model){
        try {
            ExcelImportUtil<Employee> excelImportUtil=new ExcelImportUtil<>();
            List<Employee> employees = excelImportUtil.
                excelImportUtil(uploadFile.getInputStream(), Employee.class, 1)
            excelService.importExcel(employees);
            model.addAttribute("result","上传成功");
        }catch (Exception e){
            e.printStackTrace();
            model.addAttribute("result","上传失败");
        }
        return "uploadsuccess";
}
```