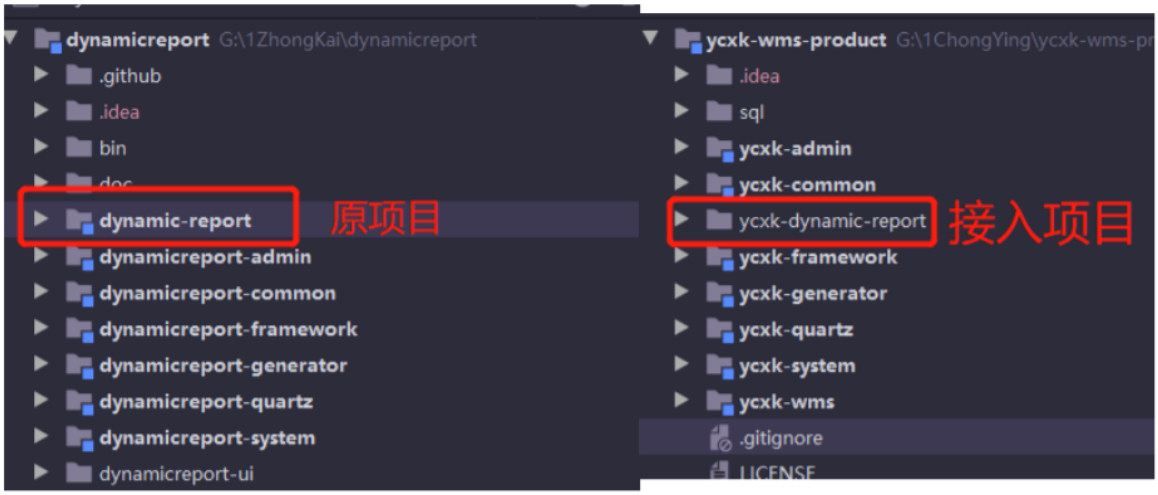
动态报表接入系统步骤

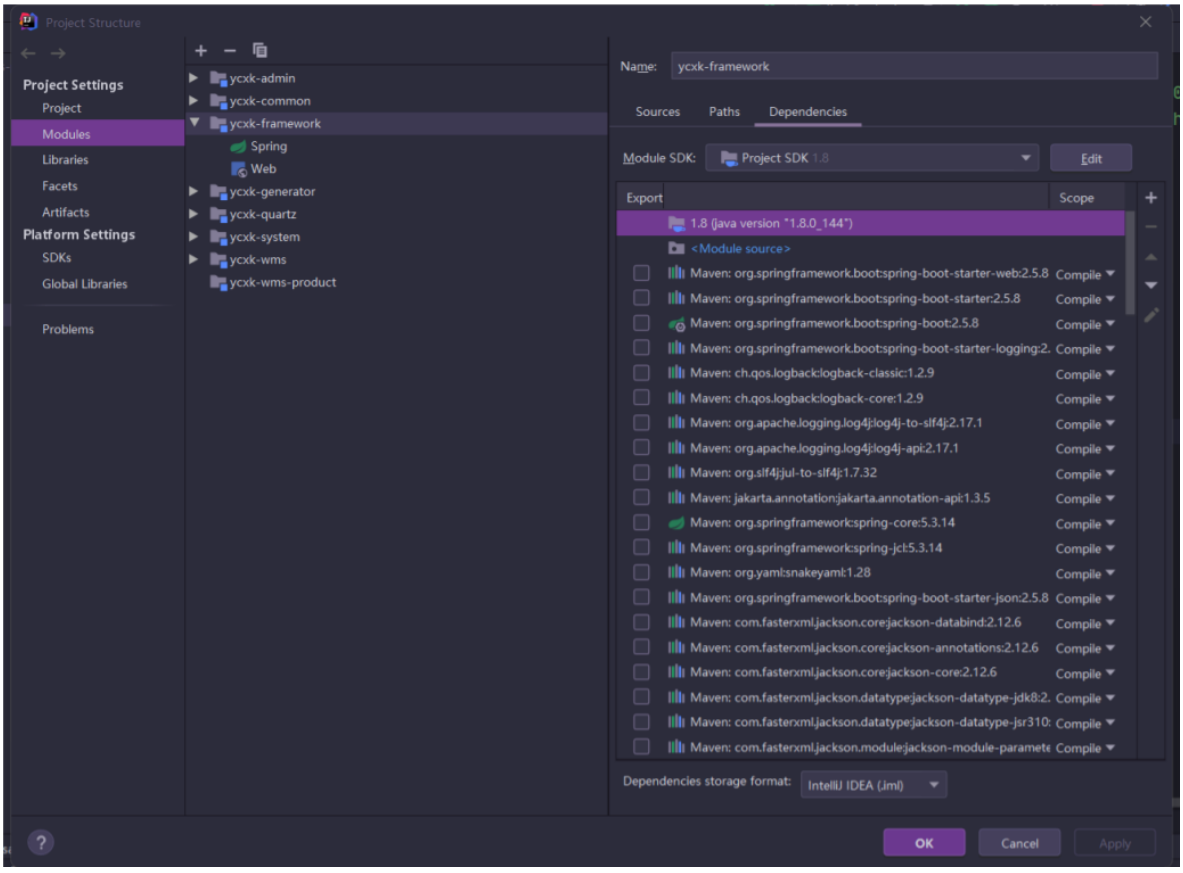
01-后端项目接入

注：步骤使用idea开发工具，框架统一使用的是ruoyi前后分离项目，由于ruoyi的common基本为通 用，只需少量修改，我们只需要把动态报表模块拉进来，把common模块增加一小部分代码即可实现后 端接入

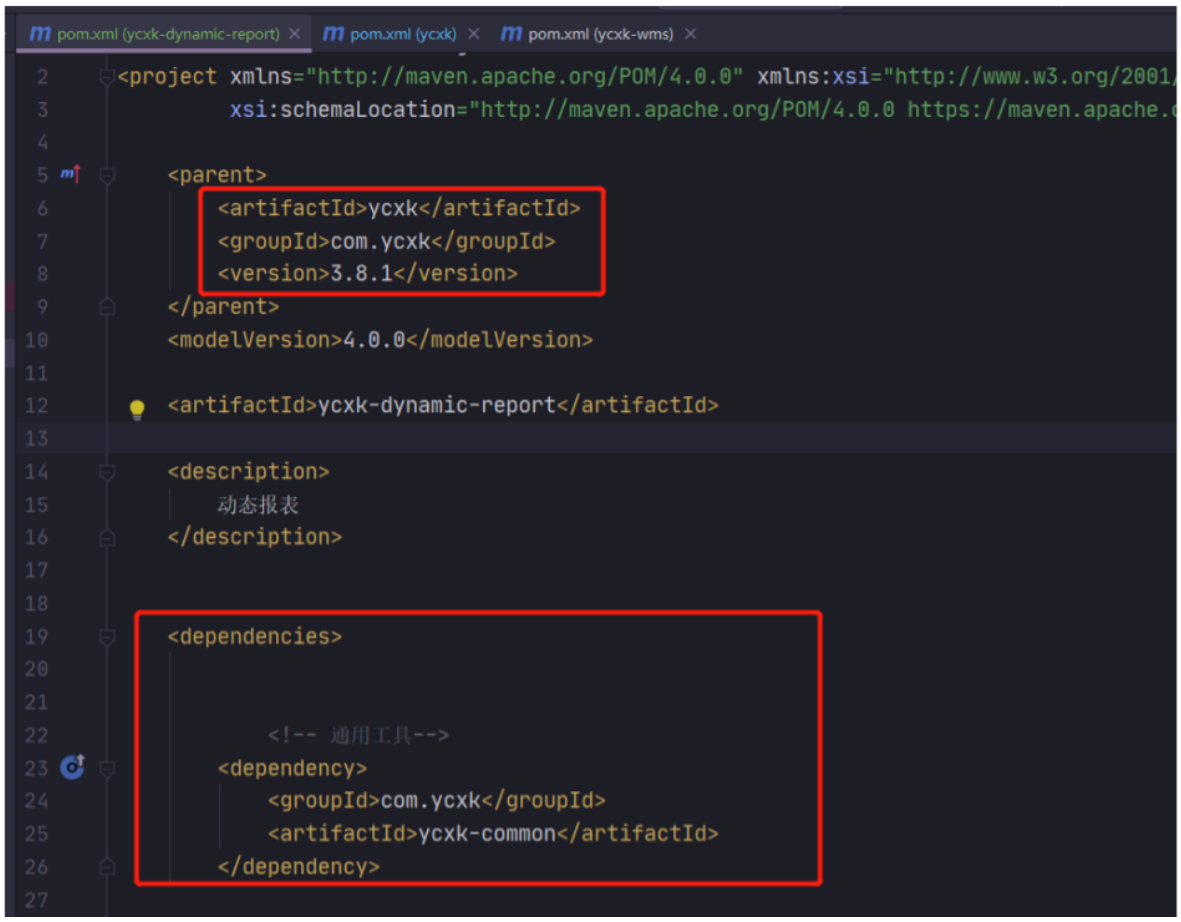
1. 复制源项目模块到接入系统，并修改项目前缀使其统一



2. 在idea-选择---File---Project Structure 打开Project Structure 窗口，打开如下如所示



3. 在打开的窗口选择modules， 点击”+“号，选择import Module， 会打开选择模块的窗口
4. 选择我们上面粘贴进来的模块----点击OK ----进入导入模块窗口----选择maven项目----确定
5. 修改pom文件 修改父级项目信息，修改依赖为本项目依赖



6. 修改顶级pom文件 修改内容如下（ hutool-core 和 ibeetl 如果在原项目中已存在，可以不加入，避免依赖冲突)

```
<dependency>
<groupId>com.ycxk</groupId>
<artifactId>ycxk-dynamic-report</artifactId>
<version>${ycxk.version}</version>
</dependency>
<dependency>
<groupId>cn.hutool</groupId>
<artifactId>hutool-core</artifactId>
<version>4.3.1</version>
</dependency>
<dependency>
<groupId>com.ibeetl</groupId>
<artifactId>beetl</artifactId>
<version>3.13.0.RELEASE</version>
</dependency>
```

7. 修改common的 pom文件 增加内容如下

```
<dependency>
<groupId>cn.hutool</groupId>
<artifactId>hutool-core</artifactId>
</dependency>
<dependency>
<groupId>com.ibeetl</groupId>
<artifactId>beetl</artifactId>
</dependency>
```

8. 在启动入口项目的pom文件下 增加如下内容

```
<dependency>
<groupId>com.ycxk</groupId>
<artifactId>ycxk-dynamic-report</artifactId>
</dependency>
```

9. 在common中的TableDataInfo类中加入 `private List sorts;` 属性，并设置对应属性方法

10. . 在common中的BaseController类中加入

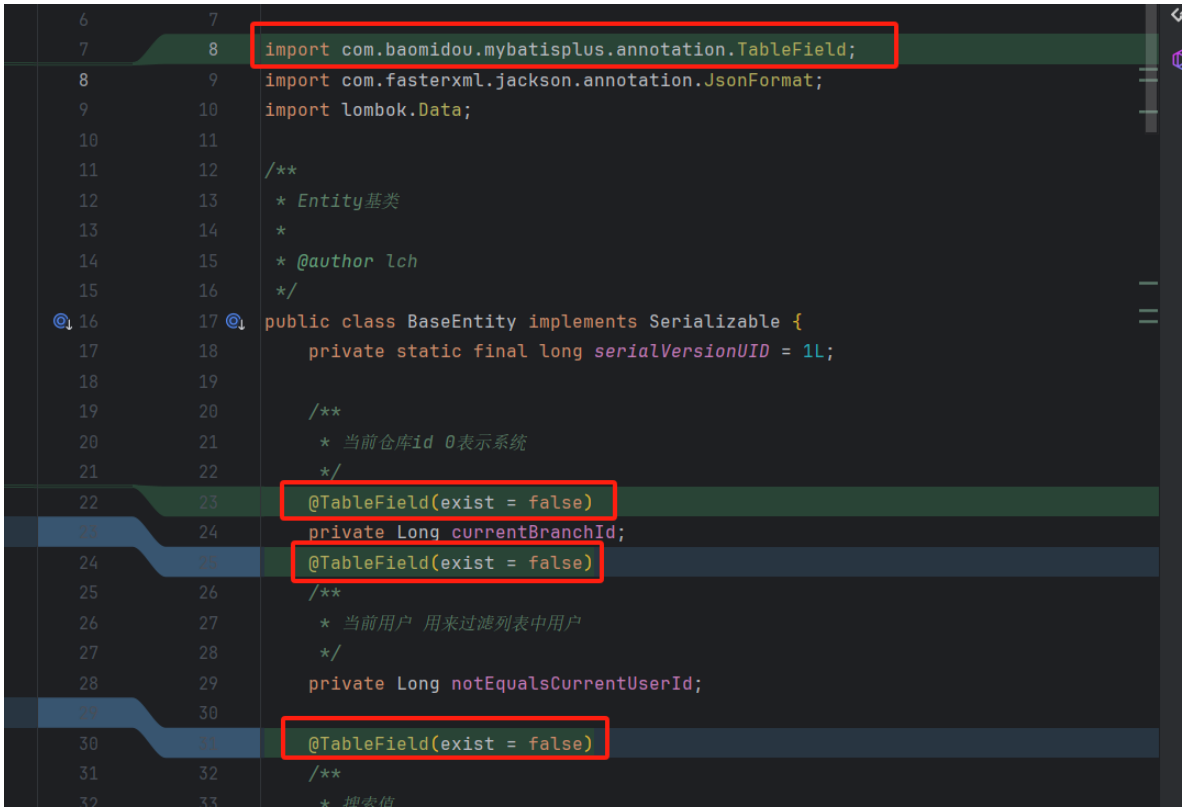
```
protected TableDataInfo getDataTable(List<?> list,List<?> sortList)
{
TableDataInfo rspData = new TableDataInfo();
rspData.setCode(HttpStatus.SUCCESS);
rspData.setMsg("查询成功");
rspData.setRows(list);
rspData.setSorts(sortList);
rspData.setTotal(new PageInfo(list).getTotal());
return rspData;
}
```

11. 在common中的SecurityUtils类中加入 ,BeanUtil包路径为cn.hutool.core.bean

```
/**
* 获取用户信息集合
* @return
*/
public static Map<String,Object> getUserInfo(){
try
{
HashMap<String, Object> userMap = new HashMap<>();
SysUser user = getLoginUser().getUser();
BeanUtil.copyProperties(user, userMap);
return userMap;
}
```

```
catch (Exception e)
{
throw new ServiceException("获取用户ID异常",
HttpStatus.UNAUTHORIZED);
}
}
```

12. 将common/core.domain 中的BaseEntity进行修改，如下图



```
import com.baomidou.mybatisplus.annotation.TableField;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;

/**
 * Entity基类
 *
 * @author lch
 */
public class BaseEntity implements Serializable {
    private static final long serialVersionUID = 1L;

    /**
     * 当前仓库id 0表示系统
     */
    @TableField(exist = false)
    private Long currentBranchId;
    @TableField(exist = false)
    /**
     * 当前用户 用来过滤列表中用户
     */
    private Long notEqualsCurrentUserId;

    @TableField(exist = false)
    /**
```

13. 在common/util/poi 下的ExcelUtil 中增加如下方法

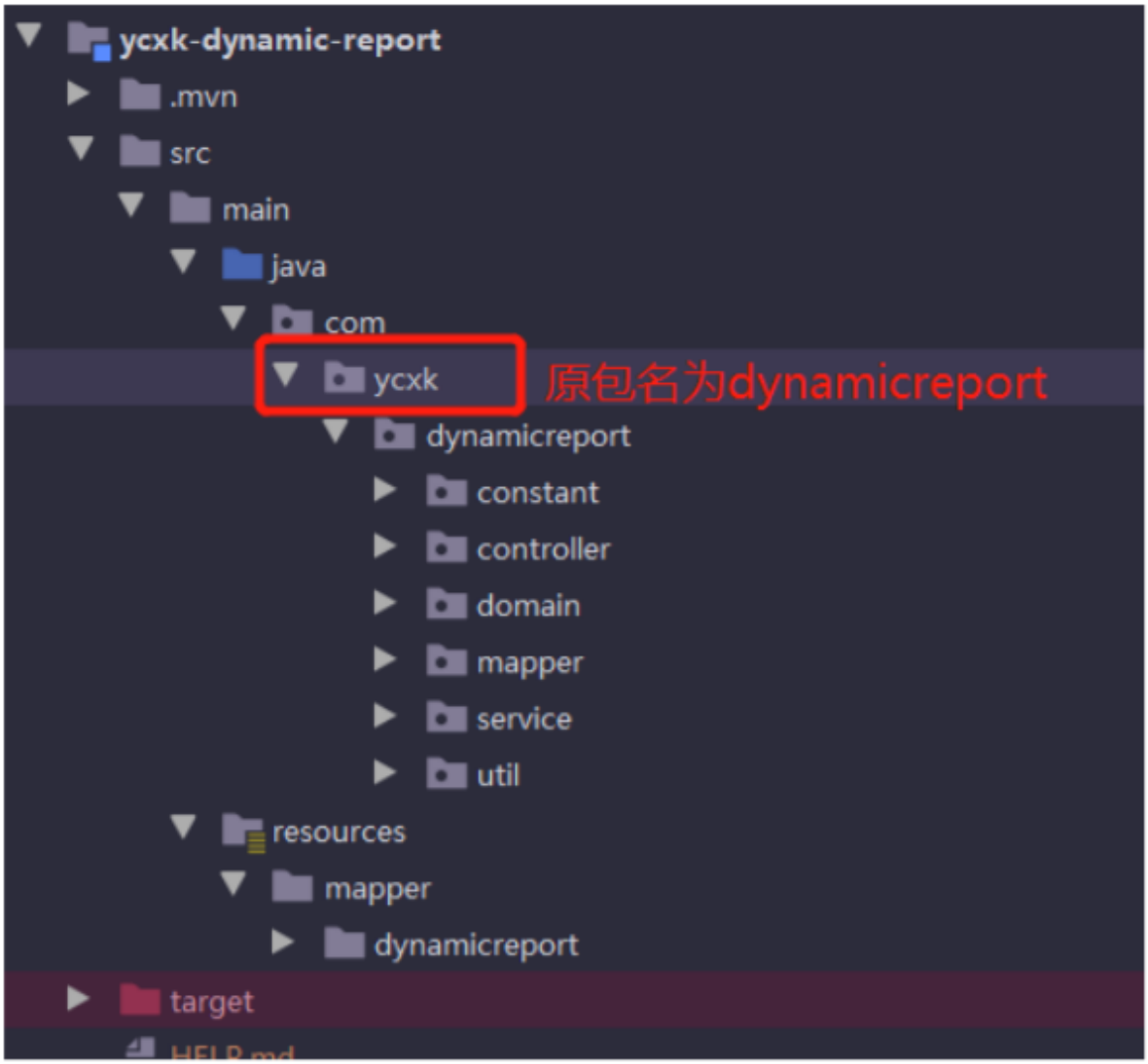
```
 //导出类型
    public static void exportNoModel(HttpServletResponse response, String []
headMap, String fileName, List<List<Object>> dataList ){
        try {
            response.setContentType("application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
            response.setCharacterEncoding("utf-8");
            String name = URLEncoder.encode(fileName, "UTF-8");
            response.setHeader("Content-disposition", "attachment;filename=" +
name + ".xlsx");
            EasyExcel.write(response.getOutputStream())
                    .head(createdHead(headMap))
                    .registerWriteHandler(new
SimpleColumnWidthStyleStrategy(15))
//                    .registerConverter(new TimestampStringConverter())
                    .registerConverter(new LocalDateStringConverter())//适配sql返
回结果为Date类型的数据导出
                    .sheet(fileName)
```

```
                    .doWrite(dataList);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

14. 在上面ExcelUtil同级目录下增加类，代码如下（此为mysql针对Date类型导出excel的解析器）

```
package com.ycxk.common.utils.poi;


import com.alibaba.excel.converters.Converter;
import com.alibaba.excel.enums.CellDataTypeEnum;
import com.alibaba.excel.metadata.GlobalConfiguration;
import com.alibaba.excel.metadata.data.WriteCellData;
import com.alibaba.excel.metadata.property.ExcelContentProperty;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.sql.Date;



public class LocalDateStringConverter implements Converter<Date> {
    @Override
    public Class supportJavaTypeKey() {
        return Date.class;
    }



    @Override
    public CellDataTypeEnum supportExcelTypeKey() {
        return CellDataTypeEnum.STRING;
    }

    @Override
    public WriteCellData<?> convertToExcelData(Date localDate,
ExcelContentProperty excelContentProperty, GlobalConfiguration
globalConfiguration) throws Exception {
        WriteCellData cellData = new WriteCellData();
        cellData.setType(CellDataTypeEnum.STRING);
        cellData.setStringValue(localDate.toString());
        cellData.setData(localDate);
        return cellData;
    }
}
```

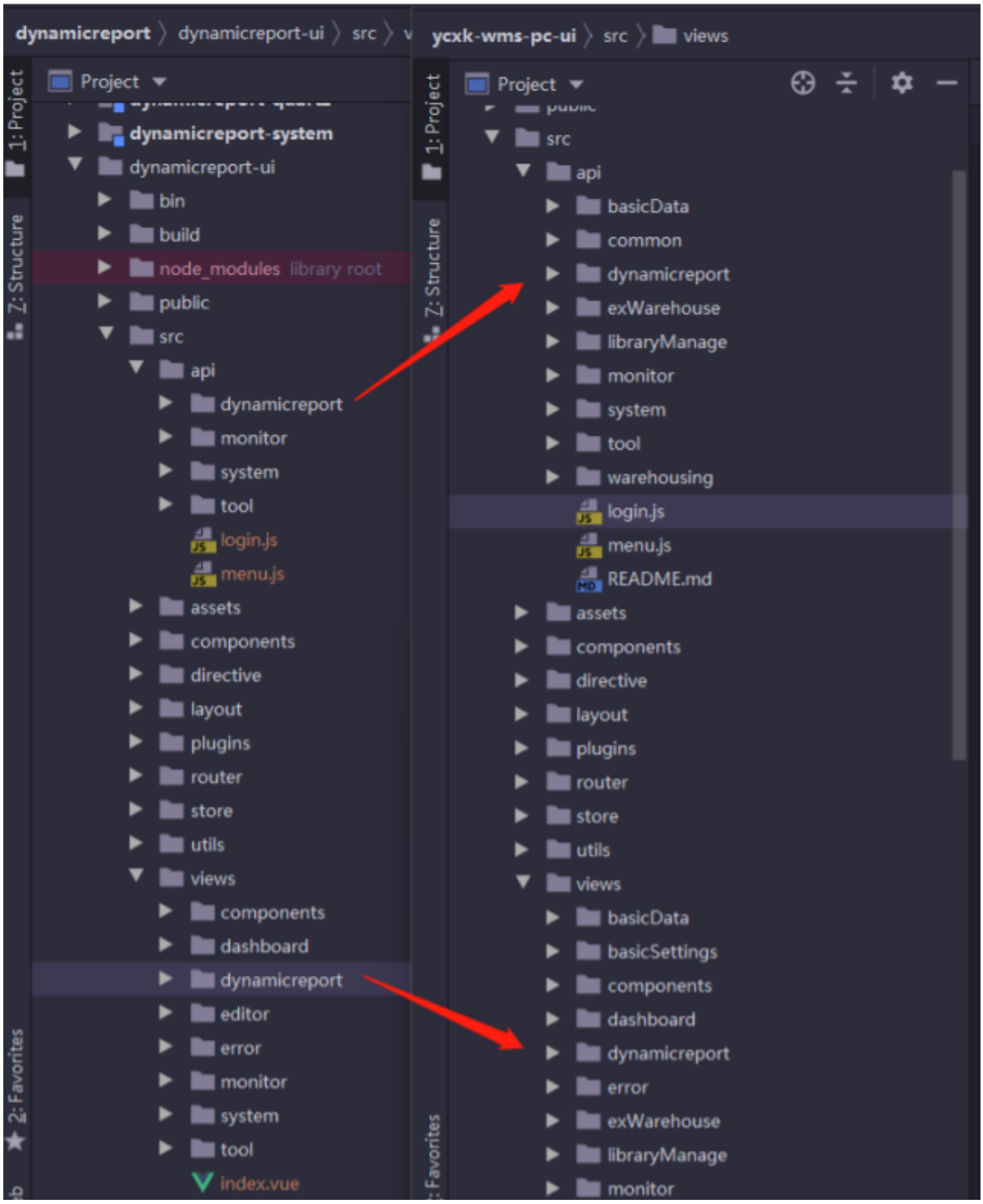
15. 修改新导入动态报表模块包路径和项目接口包路径一直 此步骤将原来的包名为dynamicreport 改 为了ycxk,如下



16. 由于新导入的动态报表模块依赖的common切换到本项目的common依赖，将代码中涉及到原common的代码修改为现在修改的common路径,不然找不到文件，可以用idea的全局搜索替换实 现 快捷键 ctrl+Shift+R 如下，可进行逐步替换，或者全部替换，这一步要看仔细，不要替换错了

02-前端项目接入

1. 分别把view和api下的dynamicreport文件夹copy到需要导入的前端项目对应的view和api下，如下 图



2. 因为报表平台涉及到在线编辑器monaco-editor，需要额外安装monaco-editor依赖

3. 在导入项目中打开控制台，在控制台输入

```
npm install monaco-editor@0.30.1 --save
```

```
npm install monaco-editor-webpack-plugin@6.0.0 --save
```

4. 安装完之后在vue.config.js中增加以下代码

```
const MonacoWebpackPlugin = require('monaco-editor-webpack-plugin')
module.exports = {
configureWebpack: {
plugins: [
new MonacoWebpackPlugin({
languages:["mysql"],
features:["coreCommands","find"]
})
]
}
}
```