

用友BIP | iuap平台

YonBuilder 专业服务开发培训

YPD 专业版脚手架扩展开发

数智平台事业部

2023 年 3 月

# 文档修订摘要

| 日期         | 修订号          | 描述                | 著者    | 审阅者 |
|------------|--------------|-------------------|-------|-----|
| 2022-09-15 | V830         | 创建                | 【卢鹏菲】 |     |
| 2022-09-26 | V83020220926 | 添加冻结模态框功能实现       | 【孙超】  |     |
| 2023-03-01 | V20221230    | 添加框架图，按照1230调整案例  | 【卢鹏菲】 |     |
| 2023-05-01 | V2023R2      | 调整冻结人             | 【卢鹏菲】 |     |
| 2023-11-06 | V20230930    | 后段扩展开发补充部分后端API说明 | 【卢鹏菲】 |     |
|            |              |                   |       |     |
|            |              |                   |       |     |
|            |              |                   |       |     |
|            |              |                   |       |     |
|            |              |                   |       |     |

# 版权

©2023 用友集团版权所有。

未经用友集团的书面许可，本文档描述任何整体或部分的内容不得被复制、复印、翻译或缩减以用于任何目的。本文档描述的内容在未经通知的情形下可能会发生改变，敬请留意。请注意：本文档描述的内容并不代表用友集团所做的承诺。

数智平台事业部

## 目录

|                          |           |
|--------------------------|-----------|
| 版权.....                  | 2         |
| <b>1 业务场景.....</b>       | <b>4</b>  |
| <b>2 前端脚本扩展开发.....</b>   | <b>5</b>  |
| 2.1 业务场景 .....           | 5         |
| 2.2 功能开发 .....           | 5         |
| 2.2.1 单据扩展案例.....        | 5         |
| 2.2.2 查询区扩展说明 .....      | 9         |
| <b>3 YPD 后端扩展开发.....</b> | <b>12</b> |
| 3.1 YPD 后端脚手架工程.....     | 12        |
| 3.2 YPD 开发框架.....        | 12        |
| 3.2.1 后端框架图.....         | 12        |
| 3.2.2 常用 API.....        | 15        |
| 3.3 插件扩展开发示例 .....       | 16        |
| 3.3.1 插件扩展开发 .....       | 16        |
| 3.3.2 业务场景 .....         | 18        |
| 3.3.3 功能开发 .....         | 18        |
| 3.3.4 应用效果 .....         | 27        |
| <b>4 前后端调用扩展示例 .....</b> | <b>28</b> |
| 4.1 业务需求 .....           | 28        |
| 4.2 功能开发思路 .....         | 28        |
| 4.3 按钮功能开发 .....         | 28        |
| 4.3.1 界面配置按钮 .....       | 28        |
| 4.3.2 添加模态框.....         | 29        |
| 4.3.3 前端功能开发 .....       | 32        |
| 4.3.4 后端功能开发 .....       | 36        |
| 4.4 应用效果 .....           | 41        |

# 1 业务场景

在本文档中，我们将核心演示如何通过脚手架通过编写前后端代码，完成业务相关的核心功能扩展。

本案例将以销售订单为例，销售订单【一主多子】类型节点，核心业务功能说明如下：

- 1、支持编码规则；
  - 2、支持交易类型，支持基于交易类型定制 workflow、业务流；
- 界面样式，点击页面时，将展现列表界面

The screenshot displays the iuap platform interface for managing sales orders. It includes a search bar at the top, a table of sales orders, and a detailed form for a specific order.

**Table 1: Sales Order List**

| 销售组织   | 编码                 | 销售区域 | 订货日期       | 客户   | 客户等级评分 | 开票客户 | 销售业务员 | 运输方式 | 销售部门   |
|--------|--------------------|------|------------|------|--------|------|-------|------|--------|
| 创新销售中心 | PO-20220108-000012 | 苏杭区域 | 2022-01-08 | 联想   | 3      | 联想   | mary  | 公路运输 | 销售二部   |
| 创新销售中心 | PO-20220108-000011 | 东部区域 | 2022-01-08 | 龙城地产 | 3      | 龙城地产 | joy   | 公路运输 | 创新销售中心 |
| 创新销售中心 | PO-20220106-000001 | 东部区域 | 2022-01-06 | 龙城地产 | 3      | 龙城地产 | --    | 公路运输 | 创新销售中心 |
| 创新销售中心 | PO-20220106-000002 | 东部区域 | 2022-01-06 | 龙城地产 | 3      | 龙城地产 | --    | 公路运输 | 创新销售中心 |
| 创新销售中心 | PO-20220108-000009 | 东部区域 | 2022-01-08 | 龙城地产 | 3      | 龙城地产 | --    | 公路运输 | 创新销售中心 |
| 创新销售中心 | PO-20220108-000010 | 东部区域 | 2022-01-08 | 龙城地产 | 3      | 龙城地产 | joy   | 公路运输 | 创新销售中心 |
| 合计     |                    |      |            |      |        |      |       |      |        |

**Table 2: Order Details (PO-20220110-000013)**

| 序号 | 商品编码     | 商品名称 | 单位 | 单品折扣 | 发货库存组织 | 计划发货日期 | 要求收货日期 | 结算财务组织 | 数量    | 税额 |
|----|----------|------|----|------|--------|--------|--------|--------|-------|----|
| 1  | 01000001 | 台式机  | 台  | --   | 库管中心   | --     | --     | 数智创新企业 | 1,000 | -- |
| 合计 |          |      |    |      |        |        |        |        | 1,000 |    |

**Form Fields:**

- 销售信息:** 销售组织, 交易类型, 销售业务员, 总数量, 冻结人, 备注, 流程名称, 流程版本, 单据状态 (开立志).
- 客户信息:** 客户, 客户等级评分 (3星), 开票客户, 客户网址, 未能生成二维码.
- 支付结算信息:** 结算方式 (银行转账), 收款协议, 实际收款.

我们将通过扩展开发完成：

- 1、前端监听扩展：设置默认值、配置前端编辑监听、参照扩展、UI 控件扩展；
- 2、后端扩展：保存赋值、调用后台服务、添加按钮服务；
- 3、前后端代码调试；
- 4、设计器添加按钮，并调用前后端服务代码完成功能；

## 2 前端脚本扩展开发

### 2.1 业务场景

- 1、子表录入：数量\*单价\*单品折扣\*税率=税额；数量\*单价\*单品折扣+税额=含税总价；
- 2、主表：主表含税总价=子表含税总价合计；主表数量=子表数量合计；
- 3、子表默认计算：要求收货日期=计划发货日期+5 天；

### 2.2 功能开发

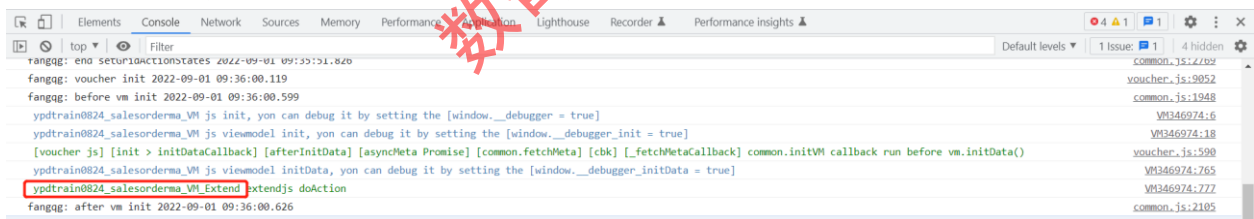
#### 2.2.1 单据扩展案例

##### 2.2.1.1 脚本接入

###### 1. 扩展脚本命名

- 1) 扩展文件名字是由应用编码（配置 cSubId，如：ypdtrain0824）和当前的单据号 billNo（配置 cBillNo，如：salesorderma）以及固定扩展字符（VM.Extend.js）组成，中间用下划线（\_）进行连接，例如：ypdtrain0824\_salesorderma\_VM\_Extend.js。

需要扩展对应页面的脚本，则可以直接打开预览页面，在 Console 控制台找到对应页面扩展文件的提示信息，按照提示信息新建文件即可。见下图



则此时创建的脚本名称为“ypdtrain0824\_salesorderma\_VM\_Extend.Extend.js”；

扩展文件放在 src/business/ {应用编码} 中。business 文件夹根据应用划分，因此在 business 下建立自己应用编码的文件夹，并将所有扩展脚本放置里面。

###### 2. 扩展脚本的使用

node 层在生成元数据模板时会加载扩展脚本，并执行 init 方法，加载业务的扩展逻辑。具体包括下面几部分：

define 参数说明：加载依赖的公共扩展脚本

doAction：框架中加载扩展脚本的入口

init：初始化整个扩展脚本

```
cb.define(process.env.__DOMAINKEY__, [], function(common) {
```

```

var emm_emm_deleaset_card_VM_Extend = {
  doAction: function(name, viewModel) {
    if (this[name]) {
      this[name](viewModel);
    }
  },
  init: function(viewModel) {
    // 初始化页面

  },
};
try {
  module.exports = emm_emm_deleaset_card_VM_Extend;
} catch (error) {
  console.log(error);
}
return emm_emm_deleaset_card_VM_Extend;
});

```

在扩展脚本中，可以获取页面控件层面的 `viewModel`，调用相关 `viewModel` 的 API 实现具体的特定的交互。可以通过相关 `api` 获取当前页面 UI 组件所对应的 `viewModel` 对象，通常扩展业务逻辑一般都写在 `init` 函数内。

### 3. 租户扩展

MDF 框架还支持不同租户自定义扩展，租户管理员可以在自定义按钮菜单，进入配置自己的扩展脚本，这样就实现不同租户加载不同的扩展脚本。通过自定义按钮菜单配置的在 Java 端返回的协议中会多个 `extscripturls` 字段，运行是会再加载 `extscripturls` 对应的扩展脚本。

## 2.2.1.2 默认值及字段控制

### 1、新增订单状态不可编辑、默认值为“正常”；

```

// 新增订单状态不可编辑、默认值为 正常
viewModel.on('modeChange', (mode) => {
  if (mode.toLocaleLowerCase() == 'add') {
    viewModel.get('orderstate').setValue('1');
    viewModel.get('orderstate').setDisabled(true);
  }
});

```

### 2.2.1.3 值改变后监听

1、子表录入：数量\*单价\*单品折扣\*税率=税额；数量\*单价\*单品折扣+税额=含税总价；

```
let detailsModel = viewModel.get('orderdetailhxList')
// 子表录入数量、价格，自动计算金额=数量*价格，录入税费后，自动计算含税总价=金额+税费
detailsModel.on('afterCellValueChange', function (data) {
  let rowIndex = data.rowIndex;
  let num = detailsModel.getCellValue(rowIndex, 'num');
  let price = detailsModel.getCellValue(rowIndex, 'price');
  let discount = detailsModel.getCellValue(rowIndex, 'discount');
  let faxrate = detailsModel.getCellValue(rowIndex, 'faxrate');

  if (['num', 'price', 'discount', 'faxrate'].includes(data.cellName)) {
    let fax = num * price * (100 - discount) * faxrate / 10000;
    let money = num * price * (100 - discount) / 100 + fax;
    if (!isNaN(fax)) { detailsModel.setCellValue(rowIndex, 'fax', fax); }
    if (!isNaN(money)) { detailsModel.setCellValue(rowIndex, 'money', money) }
  }
})
```

2、主表：主表含税总价=子表含税总价合计；主表数量=子表数量合计；

```
detailsModel.on('afterCellValueChange', function (data) {
  let rowIndex = data.rowIndex;
  let num = detailsModel.getCellValue(rowIndex, 'num');
  let price = detailsModel.getCellValue(rowIndex, 'price');
  let discount = detailsModel.getCellValue(rowIndex, 'discount');
  let faxrate = detailsModel.getCellValue(rowIndex, 'faxrate');

  if (['num', 'price', 'discount', 'faxrate'].includes(data.cellName)) {
    let fax = num * price * (100 - discount) * faxrate / 10000;
    let money = num * price * (100 - discount) / 100 + fax;
    if (!isNaN(fax)) { detailsModel.setCellValue(rowIndex, 'fax', fax); }
    if (!isNaN(money)) { detailsModel.setCellValue(rowIndex, 'money', money) }
  }
})

let detailsData = detailsModel.getData();
// 主表合计价税总计
if (['num', 'price', 'discount', 'faxrate'].includes(data.cellName)) {
  let total = 0;
  detailsData.forEach(item => {
    if (item.money) { total += item.money }
  })
}
```



```
    });  
    viewModel.get('total').setValue(total);  
  }  
  // 主表总数量  
  if (data.cellName === 'num') {  
    let totalnum = 0;  
    detailsData.forEach(item => {  
      if (item.num) { totalnum += item.num }  
    });  
    viewModel.get('totalnum').setValue(totalnum);  
  }  
})
```

数智平台事业部

- 3、子表默认计算：要求收货日期=计划发货日期+5 天；  
继续在刚才编辑的函数添加以下代码：

```
const formatDate = function (date) {  
    let y = date.getFullYear();  
    let m = date.getMonth() + 1;  
    m = m < 10 ? '0' + m : m;  
    let d = date.getDate();  
    d = d < 10 ? ('0' + d) : d;  
    return y + '-' + m + '-' + d;  
};  
// 子表默认计算：要求收货日期=计划发货日期+5 天；  
if (data.cellName === 'plansenddate') {  
    if (data.value) {  
        let date = new Date(data.value).getTime() + 86400000 * 5;  
        detailsModel.setCellValue(rowIndex, 'receiveproductdate', formatDate(new Date(date)));  
    }  
}
```

## 2.2.2 查询区扩展说明

### 2.2.2.1 脚本接入

由于查询区的 UI 元数据不随标准协议接口返回，在单据的扩展脚本的 `viewModel` 中无法获取查询区的字段模型，因此查询区可以增加单独的扩展脚本，在扩展脚本中实现类似模板扩展脚本中的交互。查询区扩展脚本也是放在 `business` 文件夹中。

#### 1. 扩展脚本命名

查询区的扩展文件名是在 `excel` 中配置的。在 `pb_meta_filters` 表中 `behaviorObject` 字段定义当前查询区扩展文件的路径。eg: `emm/emm_emm_inspectstd_list_filterVM.Extend.js`。查询区扩展脚本也放在领域 `Code` 文件夹下。

默认规范：'领域 Code\_'+单据编号\_+'filterVM.Extend.js'。



注：查询区扩展脚本定义需加上领域 `Code` 文件夹路径。

## 2. 扩展脚本使用

使用方式跟单据扩展脚本使用方式类似（示例代码如下）。

```
cb.define([ 'common/common_list_VM.Extend.js' ], function(common) {
  var emm_emm_inspectstd_list_filterVM_Extend = {
    doAction: function(name, viewModel) {
      if (this[name]) {
        this[name](viewModel);
      }
    },
    init: function(viewModel) {
      common.initOrgFilter(viewModel, 'pk_org');
      common.initCardFilter(viewModel, 'pk_org', 'pcstdlkvos__pk_equip');
    }
  };
  try {
    module.exports = emm_emm_inspectstd_list_filterVM_Extend;
  } catch (error) {
    console.log(error);
  }
  return emm_emm_inspectstd_list_filterVM_Extend;
});
```

## 3. 租户级扩展

租户管理员可以在自定义按钮菜单，进入配置自己的扩展脚本，这样就实现不同租户加载不同的扩展脚本。通过自定义按钮菜单配置的在 Java 端返回的协议中会多个 `extscripturls` 字段，运行是会再加载 `extscripturls` 对应的扩展脚本。

**extscripturls**：存放的是额外扩展文件的服务器地址，其类型可以是字符串、字符串数组。

```
extscripturls.push('http://resources.yonyoucloud.com/packages/TestExternal.js');
```

### 2.2.2.2 参照级联过滤

业务需求：供应商银行账号根据供应商参照过滤

```
viewModel.on('afterMount', function () {
  // 获取查询区模型
  const filtervm = viewModel.getCache('FilterViewModel');
  filtervm.on('afterInit', function () {
    // 进行查询区相关扩展
    // 供应商银行账号根据供应商参照过滤
    filtervm.get("supplierbankacc").getFromModel().on('beforeBrowse', function (data) {
      let supplierId = filtervm.getAllData().supplier;
      if (!supplierId.value1) {
        cb.utils.alert("请先选择供应商！", 'error');
      }
    });
  });
});
```

```
        return false;
    }
    let condition = {
        "isExtend": true,
        simpleVOs: []
    };
    condition.simpleVOs.push({
        field: 'supplier',
        op: 'eq',
        value1: supplierId.value1
    });
    this.setFilter(condition);
});
// 如果供应商参照的值有改变，就将银行账号的值清空
filtervm.get("supplier").getFromModel().on('afterValueChange', function (data) {
    filtervm.get("supplierbankacc").getFromModel().setValue(null);
});
});
});
```

数智平台事业部

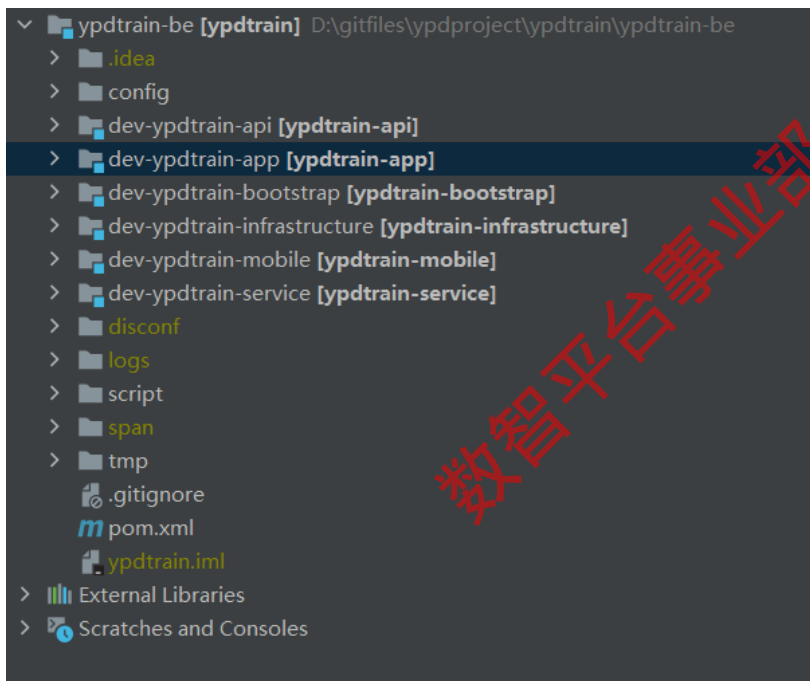
## 3 YPD 后端扩展开发

YPD 开发模式为原厂领域和项目定制开发等提供了完整的支撑工具和代码框架。YPD 框架基于业务对象，强类型对象编程，采用动作派发和插件扩展点机制实现灵活的业务代码扩展。框架提供了基础功能的最佳实践，使用方可进行灵活的业务替换和插件扩展。

### 3.1 YPD 后端脚手架工程

后端脚手架采用企业级主流框架 Springboot，对应的版本是 2.5.8

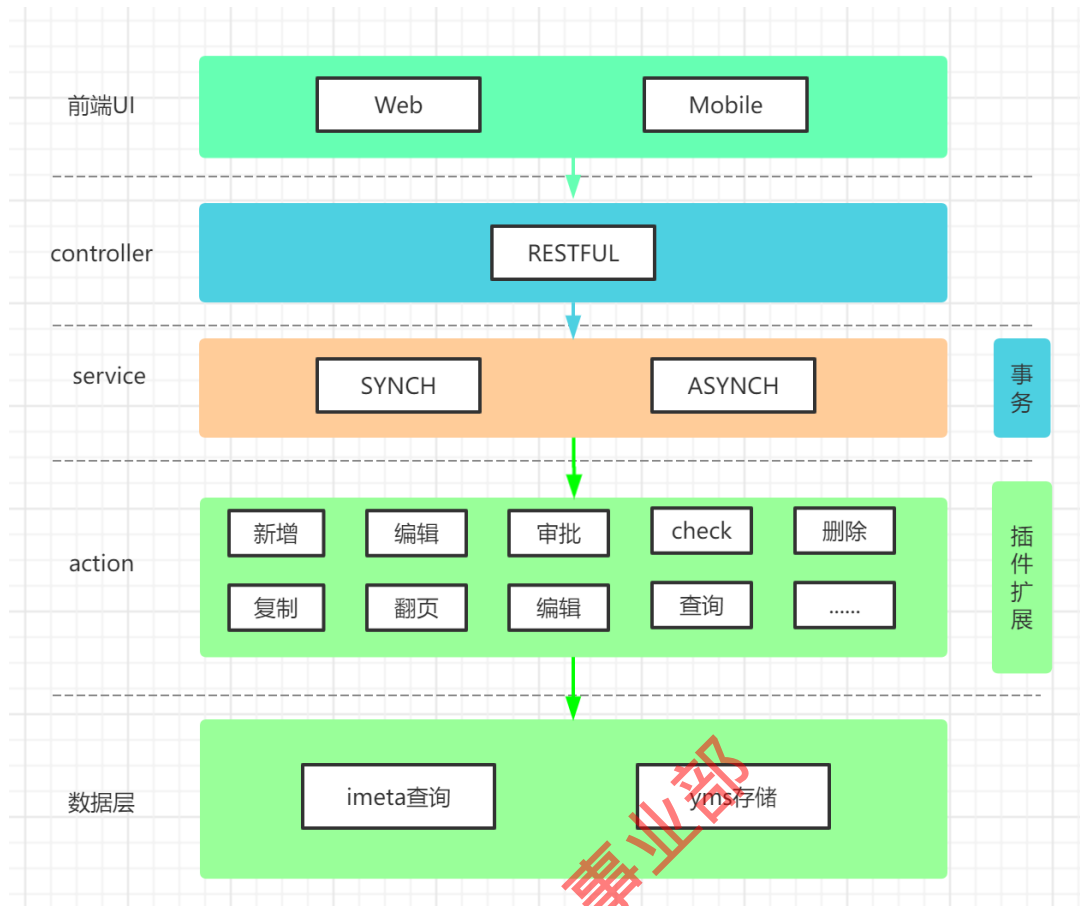
- 1、bootstrap 为项目启动模块。service 为扩展开发模块，核心业务类都放置此处。
- 2、MDFApplication 为项目启动类。
- 3、项目配置文件都在 resources 下



### 3.2 YPD 开发框架

#### 3.2.1 后端框架图

YPD 框架前后端调用框架图，通过插件扩展层，可以针对不同的动作进行业务扩展。

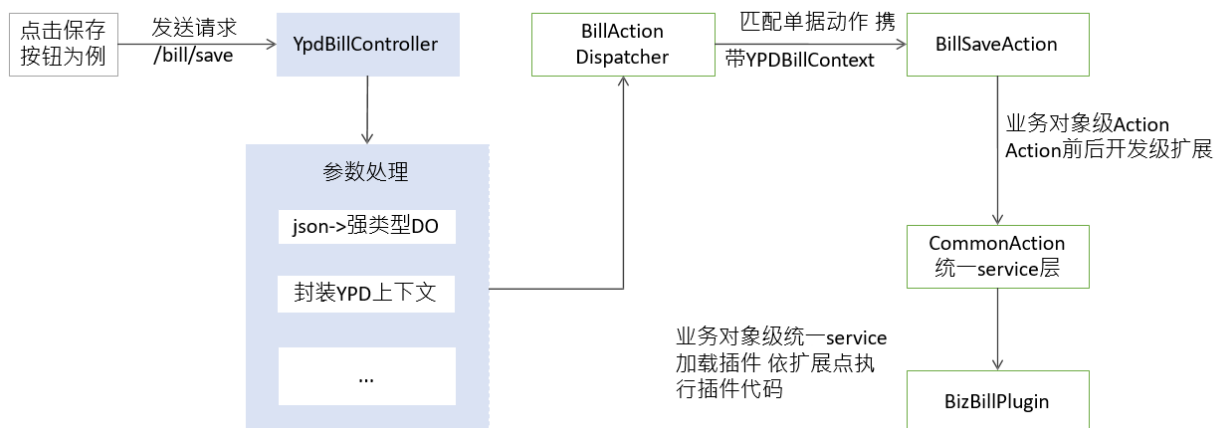


ypd 框架内置 30 个左右的单据动作实现，基本涵盖了页面上的所有操作。提供的基于策略模式和模板方法模式的抽象类（`AbsCommonBillAction`），我们可以称之为单据动作调度器。

如下图所示，`AbsCommonBillAction` 类是 action 层的统一入口，它实现了单据动作接口 `IBillBaseAction`，并扩展了基于 action 执行器级别的模板方法，即可自定义实现 `beforeExecute` 或 `afterExecute` 方法（自定义 action 时可加入自己的逻辑代码），其中 `doExecute` 是真正执行单据动作的方法。

`AbsCommonBillAction` 引入了插件执行器 `BillPluginExcutor`，表示它的子类支持了自定义插件扩展，用来支持实现复杂的业务场景。

下面以保存动作为例，动作请求链路如下图：



```

package com.yonyou.ypd.bill.controller;

import ...

@RestController
@RequestMapping("/{bill}")
public class YpdBillController {
    private static final Logger logger = LoggerFactory.getLogger(YpdBillController.class);
    @Autowired
    private IBillActionService billActionService;
    @Autowired

    @RequestMapping("/{save}")
    public JsonResult save(HttpServletRequest request, HttpServletResponse response) throws Exception {
        return this.commonBillAction( actionCode: "save", request, response);
    }

    public JsonResult commonBillAction(String actionCode, HttpServletRequest request, HttpServletResponse response) throws Exception {
        BillActionDTO billActionDto = this.billDTOCreator.createBillDataDTO(request, actionCode);
        BillActionResult actionResult = this.billActionService.executeCommonAction(billActionDto.getBaseBillContext(), billActionDto.getBillDOs(), (WorkflowParam) request.getParameterMap());
        return YpdBillResponseUtils.createCommonBillActionResult(actionResult);
    }
}

```

### BillActionServiceImpl

```

@Transactional
public BillActionResult executeCommonAction(BaseBillContext baseBillContext, IBillDO[] billDOs, WorkflowParam workflowParam) throws Exception {
    YpdBillContext billContext = new YpdBillContext(baseBillContext);
    billContext.setBillDOs(billDOs);
    billContext.setWorkflowParam(workflowParam);
    BillActionInfo actionInfo = baseBillContext.getActionInfo();
    IBillBaseAction billAction = this.loadAction(actionInfo);
    if (baseBillContext.getActionInfo().isAsync() != null && baseBillContext.getActionInfo().isAsync()) {
        return this.asyncBatchExecutor.executeAction(billAction, billContext, billDOs);
    } else {
        return billAction instanceof IBillBaseAction ? this.syncExecutor.executeAction(billAction, billContext, billDOs) : null;
    }
}

```

### 保存动作扩展示例

```

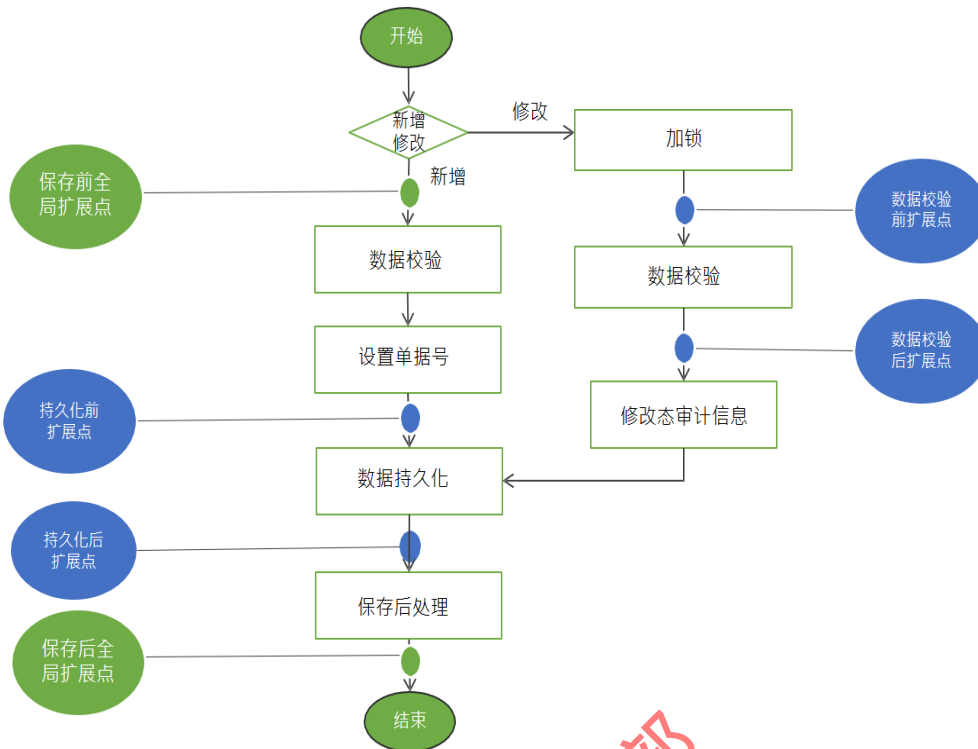
/**
 * 订单申请单据插件
 * @author wuzunqian
 * @date 2021/7/12 3:55 PM
 * 业务对象编码
 */
@BillPlugin(busiObj = "ycbpd_orderapply")
public class OrderApplyBillPlugin extends AbstractBillPlugin {

    //需求类型-庆典用烟
    private static final String VDEMANDTYPE_QDYY = "02";

    @Override
    public void initBillDefaultValueWhenNew(BpdBillContext billContext) throws Exception {
        super.initBillDefaultValueWhenNew(billContext);
        //实体类可直接强制转型
        OrderApplyDO orderApplyDO = (OrderApplyDO) billContext.getBillDO();
        //设置默认值
        orderApplyDO.setVdemandtype(VDEMANDTYPE_QDYY);
    }

    @Override
    public void beforeSave(BpdBillContext billContext) throws Exception {
        super.beforeSave(billContext);
        //汇总数量金额到表头(也可以使用公式实现)
        OrderApplyDO billDO = (OrderApplyDO) billContext.getBillDO();
        if (billDO.getYcbpd_orderapply_bList() != null && !billDO.getYcbpd_orderapply_bList().isEmpty()) {
            //获取所有表体数据
            List<OrderApplyBDO> childrenDO = BizBillDOUtils.reQueryChildrenDO(billContext.getBaseBillContext(), billDO, OrderApplyBDO.class);
            //headField->bodyField
            Map<String, String> sumFieldMap = new HashMap<>();
            sumFieldMap.put(OrderApplyDO.ISUMNUM, OrderApplyBDO.INUM);
            BizBillDOUtils.sumToHead(billDO, children);
        }
    }
}

```



为满足业务复杂多样的特点，ypd 框架会在 action 层代码固定位置预制扩展点，以增强原功能。

### 3.2.2 常用 API

1、查询接口：com.yonyou.ypd.bill.infrastructure.service.api.IBillQueryRepository

1) 查找并返回强类型，默认支持 4 层：

```
<T extends IBillDO> T findById(String metaFullName, String pk);
```

2) querySchema 查询并返回强类型：

```
List<? extends IBillDO> queryBySchema(String metaFullName, QuerySchema querySchema);
```

```
List<? extends IBillDO> queryBySchema(String metaFullName, QuerySchema querySchema, String domain);
```

```
List<? extends IBillDO> queryBySchema(String metaFullName, QuerySchema querySchema, String domain, boolean i18ndoc); // 提供是否查询多语字段，true 查询，false 不查询
```

```
Long queryTotalCount(String metaFullName, QuerySchema querySchema); // 查询总数
```

3) querySchema 查询并返回弱类型数据：

```
List<Map<String, Object>> queryMapBySchema(String metaFullName, QuerySchema querySchema);
```

4) sql 直接查询：

```
List<E> selectList(String sql, Map<String, Object> paramMap);
```

说明：sql 查询需要使用方控制 sql 拼写，这种方式无法实现穿透查询等操作,paramMap 使用 LinkedHashMap，保持 sql 中参数和 map 中顺序一致。

2、 com.yonyou.ypd.service.api.IBillCommonRepository 通用保存逻辑(不依赖 YPD 上下文)



`<T extends IBillDO> List<T> commonSaveBill(List<T> billDOs, String busiObj);`

说明：通用保存逻辑的租户用户等信息从当前上线文获取，框架补充编码和审计信息等数据，调用方建议使用此方法避免直接调用持久层。调用方需要在每个 `billDOs` 里设置 `_status` 值，支持新增（2），修改（1），删除（3），物理删除（4）

### 3、 YPD 删除逻辑

`IBillDO com.yonyou.ypd.bill.action.delete.BillDeleteProcessor.deleteBill(YpdBillContext billContext)`

### 4、将单据数据构造成树

`com.yonyou.ypd.bill.utils.BillActionUtils#buildTree`

`public static void buildTree(Pager pageData, BillEntityInfo billEntityInfo)`

示例：`BillActionUtils.buildTree(pageData, billContext.getBaseBillContext().getTreeBillEntityInfo())`

### 5、 `com.yonyou.ypd.bill.infrastructure.service.api.IbillRepository` 持久层接口

- 1) `<T> T insertBillDO(IBillDO billDO)` 新增业务数据
- 2) `MergeResultType batchInsertBillDOs(IBillDO[] billDO)` 批量新增业务数据
- 3) `int batchRemove(String fullname, List<SimpleCondition> scList)` 删除业务数据
- 4) `int batchLogicDelete(String fullname, List<SimpleCondition> scList)` 逻辑删除
- 5) `boolean updateBillDO(IBillDO billDO, String... updateFields)` 更新选定字段
- 6) `boolean batchUpdateBillDos(IBillDO[] billDO, Map<String, Set<String>> updateFieldMap)`

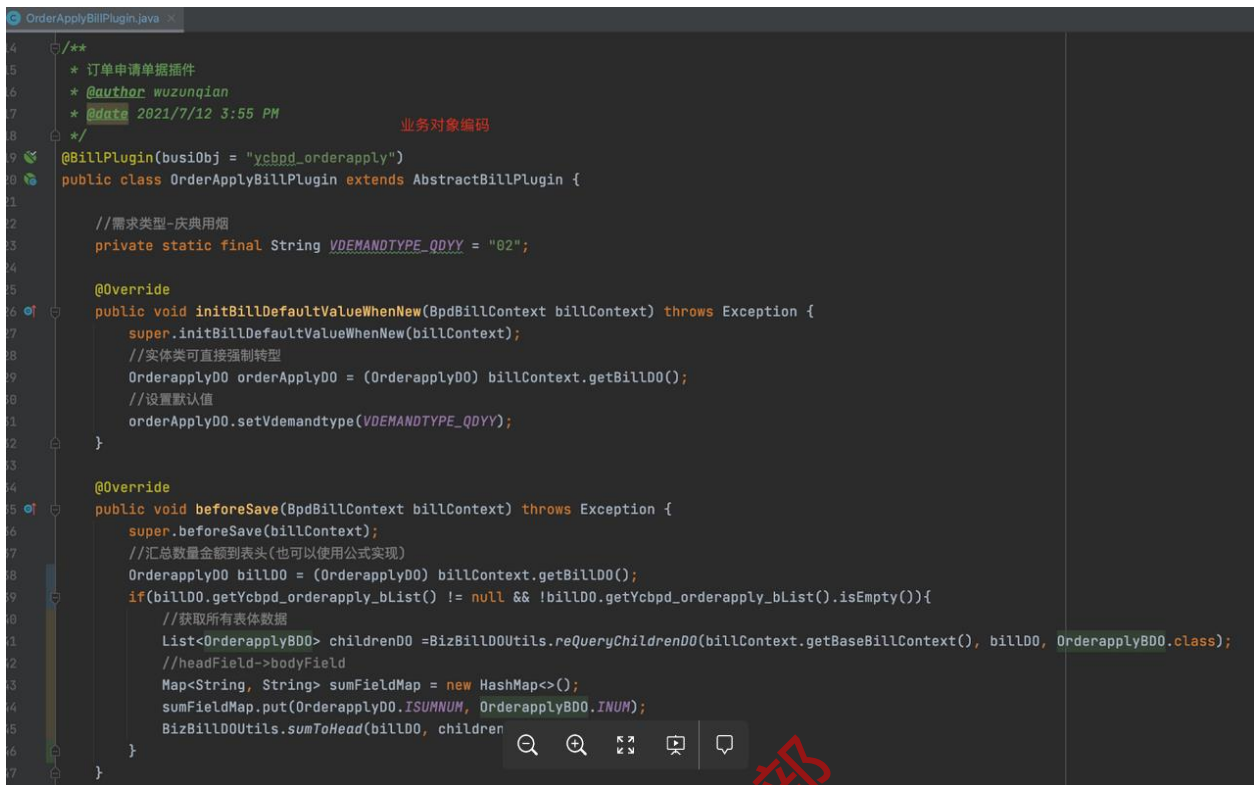
`MetaDaoHelper` 可使用

## 3.3 插件扩展开发示例

### 3.3.1 插件扩展开发

#### 1、插件开发步骤：

- 1) 创建自定义的插件 JAVA 类并继承 `com.yonyou.ypd.bill.plugin.AbstractBillPlugin` 抽象类，该抽象类实现了预制好的单据动作插件接口。
- 2) 自定义插件 JAVA 类通过 `@BillPlugin` 注解将对应的 `bean` 注入到 Spring 容器并通过其 `busiObj` 属性缓存起来，供动作调度器调用；属性 `busiObj` 就是业务对象的编码值，此处和基于业务对象的实体建模相呼应。
- 3) 根据业务需求重写抽象类对应动作的 `beforeXxxx` 或 `afterXxxx` 方法，预置插件方法如下图：



插件基类实现了所有插件接口并提供空实现，业务插件类可以按需重载。

## 2、参数上下文

单据插件类所有方法都能获取到单据上下文，单据上下文 YpdBillContext，所有的单据上下文都能获取到基本信息，像调用信息、动作编码、单据实体信息等，查询上下文额外传递了查询参数。

### ❖ YpdBillContext 参数说明

| 名称            | 类型                  | 说明        |
|---------------|---------------------|-----------|
| billDO        | IBillDO             | 单据对象（强类型） |
| billDOs       | IBillDO[]           | 单据对象数组    |
| workflowParam | WorkflowParam       | 工作流参数     |
| formulaMap    | Map<String, String> | 公式 map 集合 |

## 3、获取业务数据方法

业务对象继承于 SuperDO，且 SuperDO 实现了 IBillDO，通过控制层对数据请求参数的转化并封装到 YPD 上下文（YpdBillContext 对象实例 billContext）中，因此，在插件中可直接获取到业务对象数据，billContext.getBillDO 示例如下图

```

@Override
public void beforeSave(BpdBillContext billContext) throws Exception {
    super.beforeSave(billContext);
    // 汇总数量金额到表头 (也可以使用公式实现)
    OrderapplyDO billDO = (OrderapplyDO) billContext.getBillDO();
    if(billDO.getYcbpd_orderapply_bList() != null && !billDO.getYcbpd_orderapply_bList().isEmpty()){
        // 获取所有表体数据
        List<OrderapplyBDO> childrenDO = BizBillDOUtils.reQueryChildrenDO(billContext.getBaseBillContext(), billDO, OrderapplyBDO.class);
        // headField->bodyField
        Map<String, String> sumFieldMap = new HashMap<>();
        sumFieldMap.put(OrderapplyDO.ISUMNUM, OrderapplyBDO.INUM);
        BizBillDOUtils.sumToHead(billDO, children
    }
}

```

#### 4、实体字段赋值持久化设置方法

基于数据持久时候用到业务对象 POJO 类父类的 `Map<String, Object> beanMap` 数据，保存赋值字段到数据库，需要使用 `setAttribute` 方法将其对应字段及其赋值设置到 `beanMap` 里。

### 3.3.2 业务场景

- 1、保存前校验销售区域档案启用状态是否为“启用”；
- 2、调用业务中台基本档案提供的 RPC 服务。

组织模型创建时，可定制“销售业务委托关系”，设置销售业务主体及管控模式中常见的“委托关系”如下图。



- 1) 销售订单保存时查询“销售业务委托关系”，根据销售订单-销售组织信息，查询默认“业务委托关系”，查询到“库存组织”、“开票组织”数据；
- 2) 查询到的业务数据默认保存到子表对应的“库存组织”、“结算财务组织”字段上【字段赋值】；

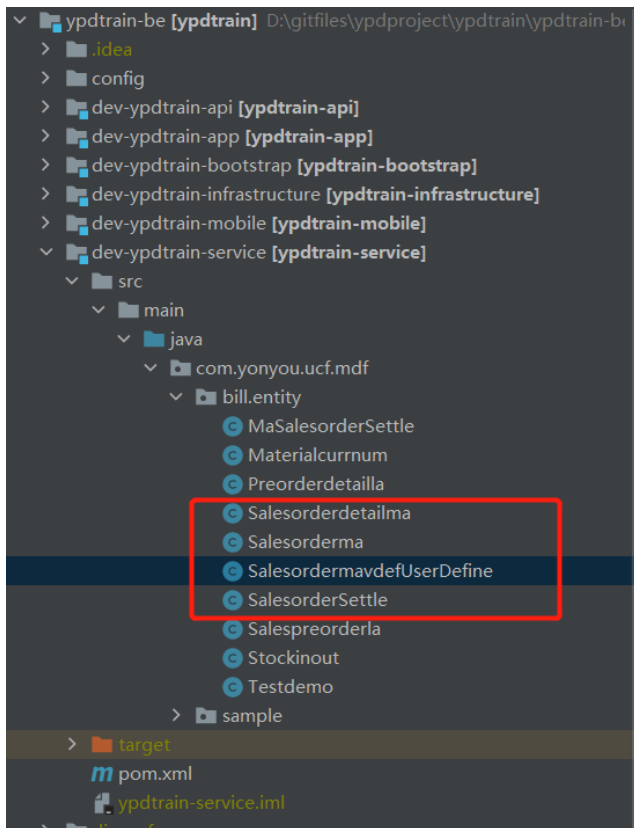
开发思路：

- 销售预单、销售订单通用规则，保存时自动根据销售业务委托关系填写组织信息；
- 使用 RPC 调用业务中台基本档案提供服务；

### 3.3.3 功能开发

#### 3.3.3.1 生成单据实体数据对象(DO)

对象建模中找到对应实体导出 `JavaBean` 或者导出 `POJO`，导出的实体类放到脚手架中。



Salesorderma 订单主实体

SalesorderSettle 订单主实体平行表-结算信息

Salesorderdetailma 订单明细实体

SalesordermavdefUserDefine 自定义特征

实体 JAVAPOJO 类集成 SuperDO 类, 注解@YMSEntity, 描述 java 实体与数据库表的映射关系, 其中 name 属性描述实体 URI, domain 属性描述域名, namespace 属性描述所属命名空间, 如下图:

```
import com.yonyou.ypd.bill.basic.entity.SuperDO;

/**
 * @author root
 * @description 销售订单ma
 * @Date 2022-08-29 15:13:45
 */
@YMSEntity(name = "ypdtrain0824.ypdtrain0824.salesorderma", domain = "ypdtrain")
public class Salesorderma extends SuperDO {
    public static final String ENTITY_NAME = "ypdtrain0824.ypdtrain0824.salesorderma";
    public static final String REVPROTOCOL = "revprotocol";
    public static final String PAYBANKACC = "paybankacc";
    public static final String REV BANKACC = "revbankacc";
    public static final String BILLNO = "billno";
    public static final String SALESORG = "sales0rg";
    public static final String SALESDEPT = "salesdept";
    public static final String SALESAREA = "salesarea";
    public static final String ORDERDATE = "orderdate";
    public static final String BUSINESSMAN = "businessman";
    public static final String SETTLETYPE = "settletype";
    public static final String CUSTOMER = "customer";
    public static final String SETTLECUST = "settlecust";
    public static final String TOTALNUM = "totalnum";
    public static final String TOTAL = "total";
    public static final String ORDERSTATE = "orderstate";
    public static final String CUSTWEBSITE = "custwebsite";
    public static final String CUSTOMERLEVEL = "customerlevel";
}
```

### 3.3.3.2 新建 plugin 插件类

- 1、新建类 com.yonyou.train.salesorder.plugin.SalesOrderPlugin
- 2、继承 AbstractBillPlugin
- 3、实现注解 @BillPlugin(busiObj = "salesorder") 对应业务对象编码;
- 4、实现方法 beforeSave()

```
package com.yonyou.train.salesorder.plugin;  
  
import com.yonyou.ypd.bill.annotation.BillPlugin;  
import com.yonyou.ypd.bill.plugin.AbstractBillPlugin;  
  
@BillPlugin(busiObj = "salesorder")  
public class SalesOrderPlugin extends AbstractBillPlugin {  
}
```

### 3.3.3.3 保存前查询数据并校验

复写 beforeSave 方法，调用查询持久层方法使用 ID 查询，查询时需要传入销售区域定义元数据 uri

```

package com.yonyou.train.salesorder.plugin;

import com.yonyou.train.salesorder.entity.Salesareasma;
import com.yonyou.ucf.mdd.ext.exceptions.BusinessException;
import com.yonyou.ucf.mdd.ext.filter.util.StringUtil;
import com.yonyou.ucf.mdf.bill.entity.Salesorder;
import com.yonyou.ypd.bill.annotation.BillPlugin;
import com.yonyou.ypd.bill.context.YpdBillContext;
import com.yonyou.ypd.bill.infrastructure.service.api.IBillQueryRepository;
import com.yonyou.ypd.bill.plugin.AbstractBillPlugin;
import org.springframework.beans.factory.annotation.Autowired;

@BillPlugin(busiObj = "salesorder")
public class SalesOrderPlugin extends AbstractBillPlugin {

    @Autowired
    private IBillQueryRepository billQueryRepository;

    @Override
    public void beforeSave(YpdBillContext billContext) throws Exception {
        super.beforeSave(billContext);
        //获取主表信息
        Salesorder salesorderma = (Salesorder) billContext.getBillDO();
        //校验合法性
        checkDataValidate(salesorderma);
    }

    /**
     * 查询校验
     *
     * @param salesorder
     * @throws Exception
     */
    private void checkDataValidate(Salesorder salesorder) throws Exception {
        String salesArea = salesorder.getSalesarea();
        if (!StringUtil.isEmpty(salesArea)) {
            //新 YPD 模式查询
            Salesareasma salesarea =
            billQueryRepository.findById("ypdtrain0824.ypdtrain0824.salesareasma", salesArea);
            if (salesarea != null) {
                String enable = salesarea.getEnable();
                String name = salesarea.getName();
            }
        }
    }
}

```

```
        if (enable.equals("0")) {
            throw new BusinessException("销售区域" + name + "已经停用， 请检查！ ");
        }
    }
}
}
```

也可以用 mdd 方法查询，如下图

```
private void checkDataValidate(Salesorderpd salesorder) throws Exception {
    String salesArea = salesorder.getSalesarea();
    if (!StringUtil.isEmpty(salesArea)) {
        BizObject result = MetaDaoHelper.findById("ypddemo.ypddemo.salesareapd", salesArea);
        if (result != null) {
            String enable = (String) result.get("enable");
            String name = (String) result.get("name");
            if (enable.equals("0")) {
                throw new BusinessException("销售区域" + name + "已经停用， 请检查！ ");
            }
        }
    }
}
```

数智平台事业部

### 3.3.3.4 调用服务查询委托关系

#### 3.3.3.4.1 新建组织 RPC 查询服务

新建类 com.yonyou.train.salesorder.service.SalesOrgRelaService

```
package com.yonyou.train.salesorder.service;

import com.yonyou.iuap.bd.common.exception.BaseDocException;
import com.yonyou.iuap.context.InvocationInfoProxy;
import com.yonyou.iuap.data.service.itf.BizDelegateApi;
import com.yonyou.iuap.org.biz.delegate.bo.BizDelegate;
import com.yonyou.iuap.org.biz.delegate.bo.SalesDelegate;
import com.yonyou.ucf.mdd.ext.exceptions.BusinessException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * 根据组织信息查询业务委托关系
 */
@Service
public class SalesOrgRelaService {

    private Logger logger = LoggerFactory.getLogger(SalesOrgRelaService.class.getName());

    @Autowired
    private BizDelegateApi bizDelegateApi;

    public Map querySalesOrgRela(String salesOrgId){
        Map orgMap = new HashMap<String, String>();
        try {
            if(salesOrgId!=null&&salesOrgId.length()>0){
                //查询业务委托关系
                List<BizDelegate> orgInfoList = bizDelegateApi.listSalesDelegateBySalesId(salesOrgId,
                    InvocationInfoProxy.getTenantid(),"diwork");
                if(orgInfoList!=null){
```



```
        for(BizDelegate bizDelegate:orgInfoList){
            SalesDelegate salesDelegate = (SalesDelegate) bizDelegate;
            Boolean isDefault = salesDelegate.getIsDefault();
            if(isDefault.booleanValue()){
                String inventOrg = salesDelegate.getInventoryOrg();
                String billingOrg = salesDelegate.getBillingOrg();
                orgMap.put("inventorg",inventOrg);
                orgMap.put("billingorg",billingOrg);

                return orgMap;
            }
        }
    }
} catch (BaseDocException e) {
    logger.error("查询委托关系异常"+e.getMessage());
    throw new BusinessException("查询业务委托关系异常");
}
return orgMap;
}
```

### 3.3.3.4.2 保存前调用

示例代码如下图所示

```
@Autowired
private SalesOrgRelaService salesOrgRelaService;

/**
 * 填写销售业务委托关系
 *
 * @param salesorder
 */
private void fillSaleOrgRela(Salesorder salesorder) {
    String salesOrg = salesorder.getSalesOrg();
    //获取子表信息
    List<Salesorderdetail> child = salesorder.getSalesorderdetailList();
    if (child != null && child.size() > 0) {
        //调用服务查询业务委托关系
        Map orgMap = salesOrgRelaService.querySalesOrgRela(salesOrg);
        if (orgMap != null && orgMap.size() > 0) {
            String stockorg = (String) orgMap.get("inventorg");
            String settleorg = (String) orgMap.get("billingorg");

            //子表数据赋值
            for (Salesorderdetail detail : child) {
                Integer voStatus = detail.get_status();
                String salesStorkOrg = detail.getStockorg();
                String salesSettleOrg = detail.getSettleorg();
                if (salesStorkOrg == null || salesStorkOrg.length() == 0){
                    detail.setStockorg(stockorg);
                    detail.setAttribute("stockorg",stockorg);
                    if(voStatus.equals(VOStatus.UPDATED)){
                        detail.addChangedPropertyNames(new String[]{"stockorg"});
                    }
                }
                if (salesSettleOrg == null || salesSettleOrg.length() == 0){
                    detail.setSettleorg(settleorg);
                    detail.setAttribute("settleorg",settleorg);
                    if(voStatus.equals(VOStatus.UPDATED)){
                        detail.addChangedPropertyNames(new String[]{"settleorg"});
                    }
                }
            }
        }
    }
}
```

```
}  
}  
}  
}
```

数智平台事业部

### 3.3.4 应用效果

录入销售组织为“创新销售中心”数据

← 返回

辅助功能

销售信息

销售组织：创新销售中心

交易类型：普通销售

销售业务员

总数量：7,000

备注：销售预订推送

流程名称：销售预订发货流程

编码：PO-20220108-000011

订货日期：2022-01-06

运输方式：公路运输

总价税合计：20,504,500.00

流程版本：V1.0

销售区域：东部区域

销售部门：创新销售中心

币种：人民币

订单状态：正常

单据状态：开立态

销售信息

客户信息

支付结算信息

保存后自动填写

← 返回

编辑 复制 提交 审批 下推 销售订单打印 打印 删除 辅助功能

总数量

备注

流程名称

客户信息

客户：联想

客户网址

支付结算信息

结算方式：银行转账

本次收款

公司信息

总价税合计

流程版本

客户等级评分：★★★★☆ 3星

收款协议

用友平台官网 iuap.yonyou.com

订单状态

单据状态：开立态

开票客户：联想

客户网址

实际收款

订单详情 订单资料 物料存量

序号

单品折扣

发货库存组织

发货日期

要求收货日期

结算财务组织

流程名称

流程版本

1

--

库管中心

--

--

数智创新企业

--

--

创建时间：2022-01-08 13:56:53

修改时间：2022-01-08 14:08:16

## 4 前后端调用扩展示例

### 4.1 业务需求

需求描述：

- 1、销售订单添加“冻结”、“解冻”按钮；
- 2、冻结功能：点击冻结时，弹出模态框，用户可以输入“冻结原因”，用户确认后，将订单状态改为“冻结”，同时后台更新冻结人、冻结日期；
- 3、解冻功能：解冻后，更新“订单状态”为正常。

### 4.2 功能开发思路

冻结功能：

- 1、冻结按钮编写控制层服务，定义/salesorder/freezebill 地址；
- 2、编写冻结功能的 Action 并实现对应方法；
- 3、前端调用/salesorder/freezebill，返回最新数据并更新界面 model

解冻功能：

- 1、解冻按钮编写控制层服务，通过/salesorder/defreezebill 方法，可以从前端调用；
- 2、后端服务层 service 实现更新功能；
- 3、前端调用/salesorder/defreezebill，返回最新数据并更新界面 model;

关闭功能：

- 1、关闭按钮编写控制层服务，通过/salesorder/defreezebill 方法，可以从前端调用；
- 2、后端服务层调用 SqlHelper 调用 mybatis 更新脚本更新业务数据；
- 3、前端调用/salesorder/defreezebill，返回最新数据并更新界面 model;

### 4.3 按钮功能开发

#### 4.3.1 界面配置按钮

添加下拉按钮“辅助功能”、添加冻结、解冻按钮，修改多语标题，同时保存。

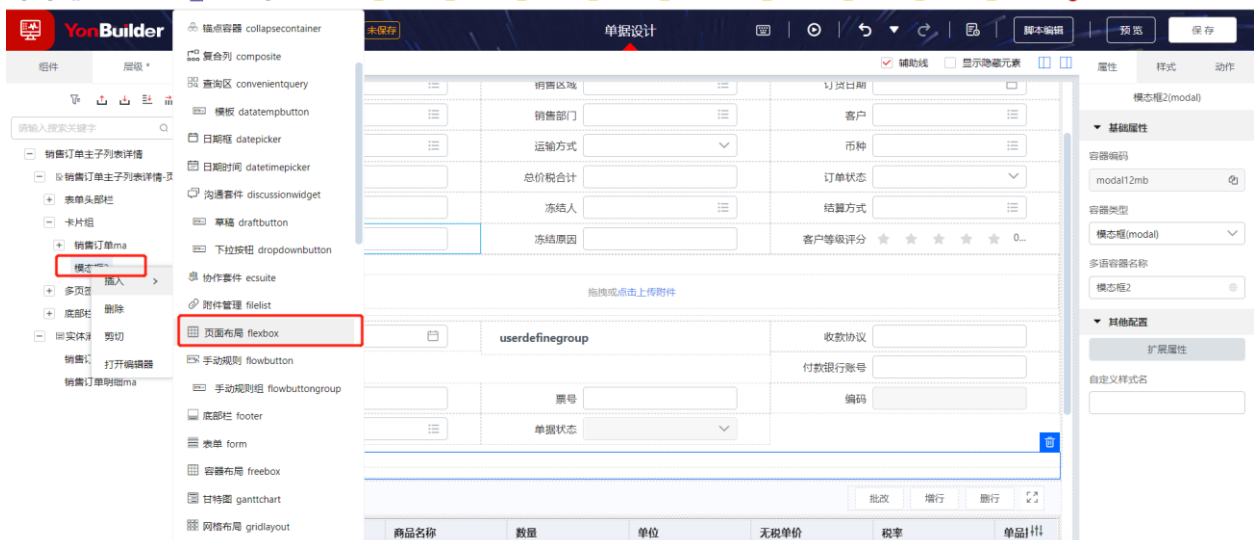


配置按钮可见属性  
冻结/解冻功能均可



## 4.3.2 添加模态框

1、页面新添模态框，右键插入“页面布局”

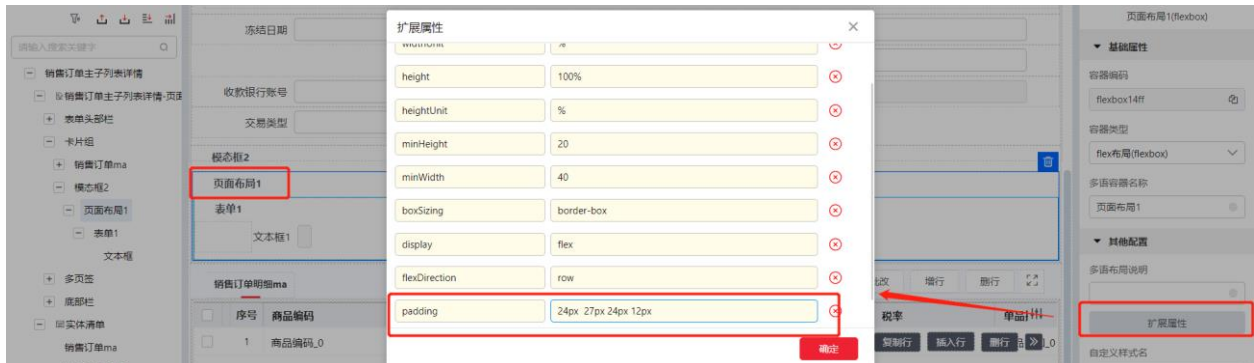


2、添加“表单”，添加“文本框”字段，名称改为“冻结原因”；

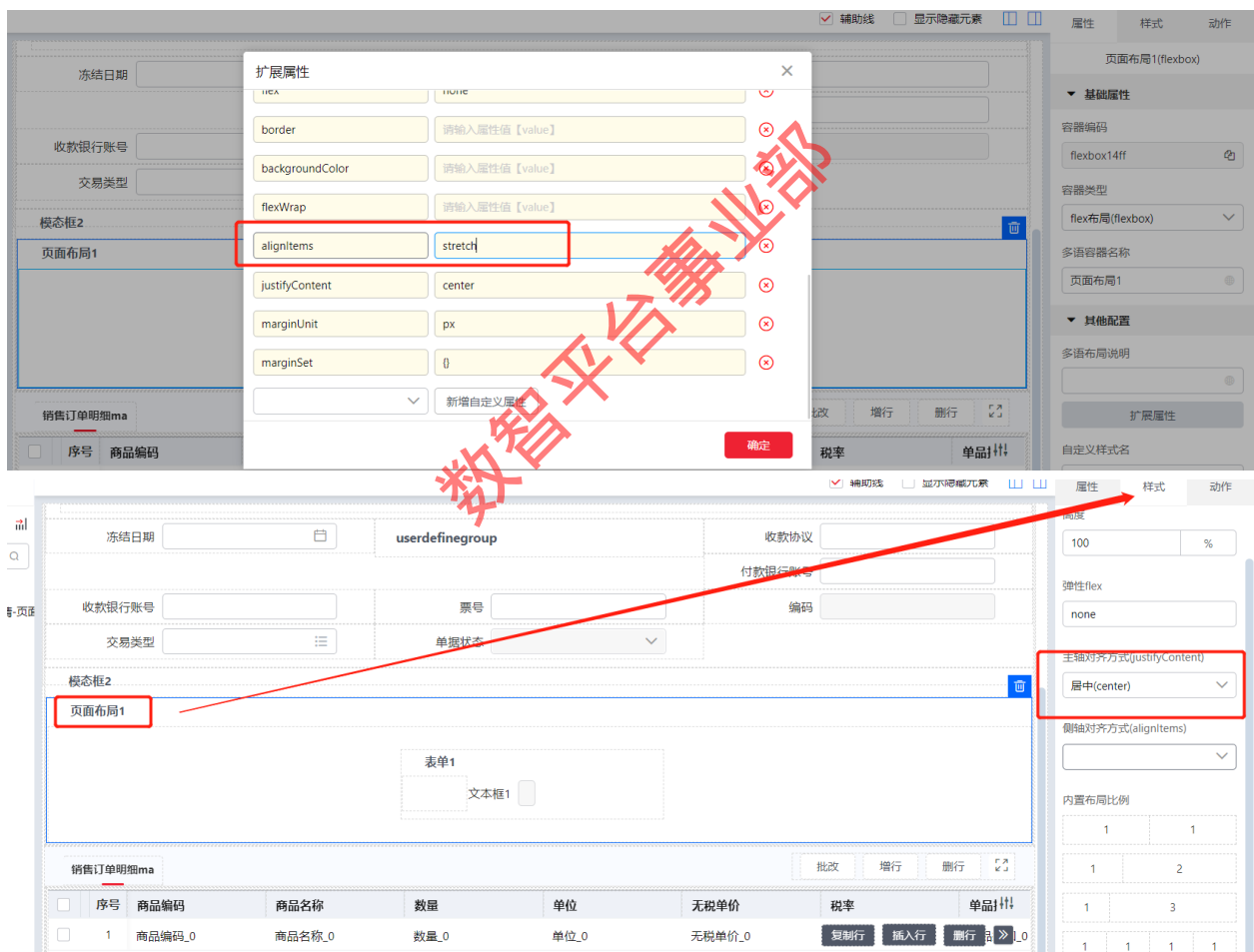
3、调整模态框样式

在页面布局添加扩展属性，如下图

Padding: 24px 27px 24px 12px



配置 alignItems 为"stretch"

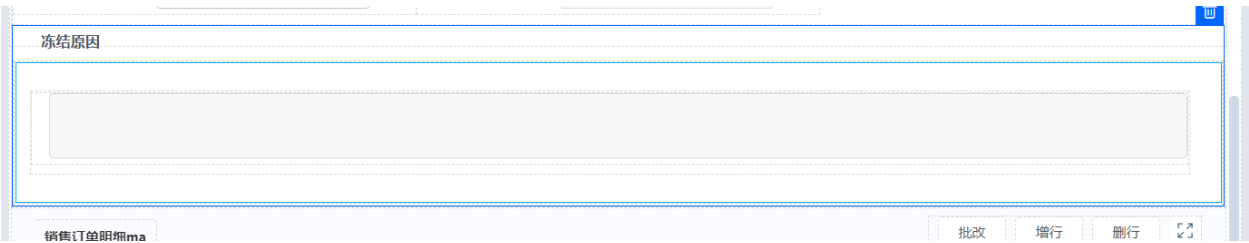




效果如下图



删除页面布局 1 显示名称、表单 1 显示名称，最终显示效果如下图





### 4.3.3 前端功能开发

1、卡片点击按钮，如果状态已经是冻结或者关闭，按钮不可用。扩展 js 如下

```
// 卡片详情页面才显示冻结、解冻按钮
viewModel.on('afterLoadData', function (data) {
  if (data.orderstate == '4') {
    viewModel.get('button29yf').setDisabled(true)
    viewModel.get('button46vf').setDisabled(false)
  } else {
    viewModel.get('button29yf').setDisabled(false)
    viewModel.get('button46vf').setDisabled(true)
  }
})
```

数智平台事业部

## 2、按钮功能实现，根据已有数据渲染界面

//冻结功能代码实现

```
// 冻结
viewModel.get('button29yf').on('click', function (data) {
  viewModel.get('item135bc').setReadOnly(false)
  viewModel.communication({
    type: "modal",
    payload: {
      mode: "inner",
      groupCode: "modal12mb", // 模态框组件的编码
      viewModel
    }
  })
})

viewModel.on('afterOkClick', args => {
  // 冻结--单击 /sales/freeze
  var proxy = cb.rest.DynamicProxy.create({
    settle: {
      url: "/salesorder/freezebill",
      method: 'POST',
      options: {}
    }
  });
  // 单据数据处理
  let data = viewModel.getAllData()
  data.reason = viewModel.get('item135bc').getValue()
  proxy.settle({
    billnum: viewModel.getParams().billNo,
    data
  }, function (err, result) {
    if (result?.error) {
      cb.utils.alert(err?.message || '冻结失败！', 'error');
    } else {
      cb.utils.alert('操作成功');
      viewModel.execute('refresh');
    }
  });
  return promise;
});
```

//解冻功能代码实现

```
// 解冻数据
viewModel.get('button46vf').on('click', function () {
    let data = viewModel.getAllData();
    var url = '/salesorder/defreezebill';
    var proxy = cb.rest.DynamicProxy.create({
        ensure: {
            url: url,
            method: 'POST'
        }
    });
    var params = {
        billnum: viewModel.getParams().billNo,
        data
    }
    proxy.ensure(params, function (err, result) {
        if (err) {
            cb.utils.alert(err.message || '解冻失败', 'error');
            return;
        }
        cb.utils.alert('解冻成功', 'success');
        viewModel.execute('refresh');
    });
})
```

```
// 关闭数据
viewModel.get('button65kh').on('click', function () {
    let data = viewModel.getAllData();
    var url = '/salesorder/closebill';
    var proxy = cb.rest.DynamicProxy.create({
        ensure: {
            url: url,
            method: 'POST'
        }
    });
    var params = {
        billnum: viewModel.getParams().billNo,
        data
    }
    proxy.ensure(params, function (err, result) {
        if (err) {
            cb.utils.alert(err.message || '关闭失败', 'error');
            return;
        }
        cb.utils.alert('关闭成功', 'success');
        viewModel.execute('refresh');
    });
})
```

## 4.3.4 后端功能开发

### 4.3.4.1 控制层代码

#### 1、新建 com.yonyou.train.salesorder.controller.SalesorderController 类

```
package com.yonyou.train.salesorder.controller;;

import com.yonyou.train.salesorder.service.SalesOrderService;
import com.yonyou.ypd.bill.action.result.BillActionResult;
import com.yonyou.ypd.bill.basic.bean.JsonResult;
import com.yonyou.ypd.bill.basic.dto.BillActionDTO;
import com.yonyou.ypd.bill.basic.service.api.IBillActionService;
import com.yonyou.ypd.bill.dto.YpdBillDTOCreator;
import com.yonyou.ypd.bill.response.YpdBillResponseUtils;
import com.yonyou.ypd.bill.workflow.param.WorkflowParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@RestController
@RequestMapping("/salesorder")
public class SalesorderController{

    @Autowired
    private YpdBillDTOCreator billDTOCreator;

    @Autowired
    private IBillActionService billActionService;

    @Autowired
    SalesOrderService salesOrderService;

    @RequestMapping({"/freezebill"})
    public JsonResult freeze(HttpServletRequest request, HttpServletResponse response) throws Exception {
        BillActionDTO billActionDto = this.billDTOCreator.createBillDataDTO(request, "freezebill");
        BillActionResult actionResult =
this.billActionService.executeCommonAction(billActionDto.getBaseBillContext(), billActionDto.getBillDOs(),
(WorkflowParam)null);
        return YpdBillResponseUtils.createCommonBillActionResult(actionResult);
    }
}
```

```

    }

    @RequestMapping("/{defreezebill}")
    public JsonResult deFreeze(HttpServletRequest request, HttpServletResponse response) throws Exception {
        BillActionDTO billActionDto = this.billDTCreator.createBillDataDTO(request, "defreeze");
        BillActionResult actionResult = salesOrderService.deFreezeBill(billActionDto.getBaseBillContext(),
        billActionDto.getBillDOs());
        return YpdBillResponseUtils.createCommonBillActionResult(actionResult);
    }

    @RequestMapping("/{closebill}")
    public JsonResult close(HttpServletRequest request, HttpServletResponse response) throws Exception {
        BillActionDTO billActionDto = this.billDTCreator.createBillDataDTO(request, "close");
        BillActionResult actionResult = salesOrderService.close(billActionDto.getBaseBillContext(),
        billActionDto.getBillDOs());
        return YpdBillResponseUtils.createCommonBillActionResult(actionResult);
    }
}

```

#### 4.3.4.2 服务层代码

- 1、冻结功能：使用 YPD 现有动作框架，编写冻结 Action 实现：
  - 1) 新建冻结 Action 类 com.yonyou.train.salesorder.service.SalesOrderFreezeAction
  - 2) 继承 AbsCommonBillAction
  - 3) 实现注解@BillAction(actionCode = "freezebill",actionName = "冻结",busiObj = "salesorderma")  
其中 freezebill 对应 Controller 提供的 actionCode，busiObj 对应业务对象

代码示例如下

```

package com.yonyou.train.salesorder.service;

import com.yonyou.iuap.context.InvocationInfoProxy;
import com.yonyou.iuap.message.platform.rpc.IMsgSendService;
import com.yonyou.ucf.mdf.bill.entity.Salesorder;
import com.yonyou.ypd.bill.action.AbsCommonBillAction;
import com.yonyou.ypd.bill.annotation.BillAction;
import com.yonyou.ypd.bill.basic.BaseBillContext;
import com.yonyou.ypd.bill.basic.entity.IBillDO;
import com.yonyou.ypd.bill.context.YpdBillContext;
import com.yonyou.ypd.bill.infrastructure.service.api.IBillQueryRepository;
import com.yonyou.ypd.bill.infrastructure.service.api.IBillRepository;

```

```
import com.yonyou.ypd.bill.lock.YpdBillLockData;
import com.yonyou.ypd.bill.lock.YpdBusinessLock;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import java.text.SimpleDateFormat;
import java.util.Date;

@Component
@BillAction(actionCode = "freezebill", actionName = "冻结", busiObj = "Salesorder")
public class SalesOrderFreezeAction extends AbsCommonBillAction<YpdBillContext, IBillDO> {

    @Autowired
    private IBillRepository billRepository;

    @Autowired
    private IBillQueryRepository billQueryRepository;

    @Autowired
    private YpdBusinessLock businessLock;

    @Autowired
    private IMsgSendService msgSendService;

    @Override
    protected IBillDO doExecute(YpdBillContext billContext, Object... args) throws Exception {
        IBillDO billDO = this.doFreezeBill(billContext);
        billContext.setBillDO(billDO);
        return billDO;
    }

    private IBillDO doFreezeBill(YpdBillContext billContext) throws Exception {
        Salesorder salesorder = (Salesorder) billContext.getBillDO();
        BaseBillContext baseBillContext = billContext.getBaseBillContext();

        String fullName = baseBillContext.getMetaFullName();
        //环境上下文
        String userId = InvocationInfoProxy.getUserid();

        //获取当前日期
        Date date = new Date();
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        salesorder.setFreezedate(dateFormat.format(date));
    }
}
```

```

        salesorder.setFreezeperson(userId);
        salesorder.setAttribute("freezeperson", userId);
        salesorder.setOrderstate("4");
        salesorder.setAttribute("orderstate", "4");
        salesorder.setAttribute("freezedate", salesorder.getFreezedate());

        salesorder.set_status(new Integer("1"));
        billRepository.updateBillDO(salesorder);

        return billQueryRepository.findById(fullName, salesorder.getId());
    }

    @Override
    protected void beforeExecute(YpdBillContext billContext, Object... args) throws Exception {
        this.businessLock.dynamicLockAndSetKeyToContext(billContext,
new
YpdBillLockData(billContext.getBillDO()));
        super.beforeExecute(billContext, args);
    }

    @Override
    protected void afterExecute(YpdBillContext billContext, iBillDO iBillDO, Object... args) throws Exception {
        super.afterExecute(billContext, iBillDO, args);
    }
}

```

3、解冻功能：调用 YPD 的持久层 API 直接更新并返回数据；  
新建服务类 com.yonyou.train.salesorder.service.SalesOrderService

```

package com.yonyou.train.salesorder.service;

import com.yonyou.ucf.mdf.bill.entity.Salesorder;
import com.yonyou.ypd.bill.action.result.BillActionResult;
import com.yonyou.ypd.bill.basic.BaseBillContext;
import com.yonyou.ypd.bill.basic.entity.iBillDO;
import com.yonyou.ypd.bill.infrastructure.service.api.iBillQueryRepository;
import com.yonyou.ypd.bill.infrastructure.service.api.iBillRepository;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import javax.transaction.Transactional;

@Service

```



```

public class SalesOrderService {

    private Logger log = LoggerFactory.getLogger(SalesOrderService.class);

    @Autowired
    private IBillQueryRepository billQueryRepository;

    @Autowired
    private IBillRepository billRepository;

    @Transactional
    public BillActionResult deFreezeBill(BaseBillContext billContext, IBillDO[] billDOs) throws Exception {
        //获取主表信息
        Salesorder salesorder = (Salesorder) billDOs[0];
        //更新数据
        salesorder.setOrderstate("1");
        salesorder.setAttribute("orderstate","1");

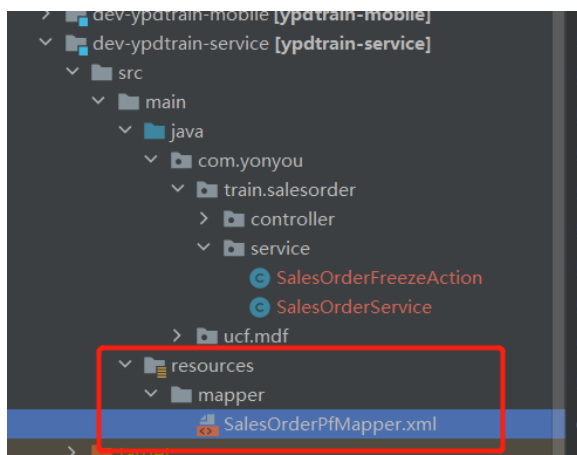
        salesorder.set_status(new Integer("1"));
        billRepository.updateBillDO(salesorder);
        Salesorder newSales = billQueryRepository.findById(billContext.getMetaFullName(),
salesorder.getId());
        //返回结果集
        BillActionResult billActionResult = new BillActionResult();
        billActionResult.setBillDO(newSales);
        return billActionResult;
    }
}

```

#### 4、关闭功能

##### 1) 添加 Mapper 文件

在 domain 下新建如下目录，注意 xml 文件应该以 Mapper.xml 结尾，如下图



## Xml 文件参考

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.yonyou.train.sales.mapper.SalesOrderMapper">

    <!--关闭功能-->
    <update id="closeOrder" parameterType="java.util.Map">
        update cc_salespreorder set orderstate='3' where id=#{id}
    </update>

</mapper>

```

## 2) Service 中调用

```

/**
 * 关闭功能
 * @param baseBillContext
 * @param billDOs
 * @return
 */
public BillActionResult close(BaseBillContext baseBillContext, IBillDO[] billDOs) throws Exception {
    //获取主表信息
    Salesorder salesorder = (Salesorder) billDOs[0];
    Map<String,Object> param = new HashMap<>();
    param.put("id",salesorder.getId());
    int count =
SqlHelper.update("com.yonyou.train.sales.mapper.SalesOrderMapper.closeOrder",param);
    Salesorder newSales = billQueryRepository.findById(baseBillContext.getMetaFullName(),
salesorder.getId());
    //返回结果集
    BillActionResult billActionResult = new BillActionResult();
    billActionResult.setBillDO(newSales);
    return billActionResult;
}

```

## 4.4 应用效果

销售订单，使用本地 local 访问，此时订单状态“正常”  
选择“辅助功能”-冻结按钮

← → ↻ ⚠ 不安全 | https://local.yonyou.com:3003/meta/VoucherList/salesorderma2List?domainKey=ypdtrain&busiObj=salesorderma&designPreview=true

000003 开立态

编辑 复制 提交 撤回 下推 全局联查 打印 删除 辅助功能 变更历史查询

销售订单ma

销售组织 销售事业部 销售区域 西部 订货日期 2022-09-13

业务员 -- 销售部门 -- 客户 --

开票客户 -- 运输方式 -- 币种 人民币

总数量 -- 总价税合计 -- 订单状态 正常

备注 -- 冻结人 -- 结算方式 --

客户网址 -- 冻结原因 -- 客户等级评分 ☆☆☆☆☆ 0星

0上传附件 (单个文件最大100MB)

拖拽或点击上传附件

冻结日期 -- 客户事项 -- 销售预测日期 --

销售客户详情 -- 收款协议 -- 付款银行账号 --

收款银行账号 -- 票号 -- 编码 000003

交易类型 -- 单据状态 开立态

销售订单明细ma

| 序号 | 商品编码 | 商品名称 | 数量       | 单位 | 无税单价 | 税率 | 单品折扣 | 价税合计 | 发货库存组织 | 计划发货日期 |
|----|------|------|----------|----|------|----|------|------|--------|--------|
| 1  | 电脑   | --   | 2,222.00 | -- | --   | -- | --   | --   | --     | --     |

点击确定后，可弹出模态框

000003 开立态

编辑 复制 提交 撤回 下推 全局联查 打印 删除 辅助功能 变更历史查询

销售订单ma

销售组织 销售事业部 销售区域 西部 订货日期 2022-09-13

业务员 -- 销售部门 -- 客户 --

开票客户 -- 运输方式 -- 币种 人民币

总数量 -- 总价税合计 -- 订单状态 正常

备注 -- 冻结人 -- 结算方式 --

客户网址 -- 冻结原因 -- 客户等级评分 ☆☆☆☆☆ 0星

0上传附件 (单个文件最大100MB)

拖拽或点击上传附件

冻结日期 -- 客户事项 -- 销售预测日期 --

销售客户详情 -- 收款协议 -- 付款银行账号 --

收款银行账号 -- 票号 -- 编码 000003

交易类型 -- 单据状态 开立态

销售订单明细ma

| 序号 | 商品编码 | 商品名称 | 数量       | 单位 | 无税单价 | 税率 | 单品折扣 | 价税合计 | 发货库存组织 | 计划发货日期 |
|----|------|------|----------|----|------|----|------|------|--------|--------|
| 1  | 电脑   | --   | 2,222.00 | -- | --   | -- | --   | --   | --     | --     |

点击确定后，订单状态更新

000003 开立态

编辑 复制 提交 撤回 下推 全局联查 打印 删除 辅助功能 变更历史查询

销售订单ma

销售组织 销售事业部 销售区域 西部 订货日期 2022-09-13

业务员 -- 销售部门 -- 客户 --

开票客户 -- 运输方式 -- 币种 人民币

总数量 -- 总价税合计 -- 订单状态 冻结

备注 -- 冻结人 -- 结算方式 --

客户网址 -- 冻结原因 -- 客户等级评分 ☆☆☆☆☆ 0星

0上传附件 (单个文件最大100MB)

拖拽或点击上传附件

冻结日期 -- 客户事项 -- 销售预测日期 --

销售客户详情 -- 收款协议 -- 付款银行账号 --

收款银行账号 -- 票号 -- 编码 000003

交易类型 -- 单据状态 开立态

销售订单明细ma

| 序号 | 商品编码 | 商品名称 | 数量       | 单位 | 无税单价 | 税率 | 单品折扣 | 价税合计 | 发货库存组织 | 计划发货日期 |
|----|------|------|----------|----|------|----|------|------|--------|--------|
| 1  | 电脑   | --   | 2,222.00 | -- | --   | -- | --   | --   | --     | --     |

点击解冻

数智平台事业部



# 用友BIP | iuap平台

数智平台事业部