**IUAP平台使用指南**

**树表组件**

用友网络科技股份有限公司

二○一八年五月

**此文档为保密文档，未经用友软件股份有限公司书面同意，不得向任何单位或个人提供、转让本文档中的任何内容，用友软件股份有限公司将保留对泄漏文档内容的起诉权利。**

**IUAP平台使用指南**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [**√**]正在修改  [] 正式发布 | 文件标识： |  |
| 当前版本： |  |
| 作者： |  |
| 完成日期： |  |

版本历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 作者 | 参与者 | 起止日期 | 备注 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

审核记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 姓名 | 职位 | 审阅日期 | 备注 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[第一章 概述 1](#_Toc515650684)

[1.1 业务场景 1](#_Toc515650685)

[1.2 开发方案 1](#_Toc515650686)

[第二章 示例配置 3](#_Toc515650687)

[2.1 数据库配置 3](#_Toc515650688)

[2.2 开发环境配置 4](#_Toc515650689)

[2.3 功能验证 5](#_Toc515650690)

[第三章 代码开发示例 9](#_Toc515650691)

[3.1 后端开发 9](#_Toc515650692)

[3.1.1 组织机构操作 9](#_Toc515650693)

[3.1.2 人员信息操作 10](#_Toc515650694)

[3.2 前端开发 10](#_Toc515650695)

[3.2.1 编写meta.js 10](#_Toc515650696)

[3.2.2 编写contacts.html文件 12](#_Toc515650697)

[3.2.3 编写contacts.css文件 13](#_Toc515650698)

[3.2.4 编写contacts.js文件 13](#_Toc515650699)

# 概述

## 业务场景

开发者经常遇到类似组织机构和人员关系的业务场景，组织机构是树状结构，人员归属于某个组织机构，典型需求如下：

* 组织机构的增加、修改、删除
* 人员的增加、修改、删除、查询

## 开发方案

* 创建业务对象

创建组织机构表(example\_organization)和联系人表(example\_contacts)。

表格 1 组织机构表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 类型 | 主键 | 备注 |
| institid | 主键 | Char(36) | 是 | Uuid |
| instit\_code | 编码 | Varchar(200) |  | 非空，唯一索引 |
| instit\_name | 名称 | Varchar(200) |  | 可空 |
| short\_name | 缩写 | Varchar(50) |  | 可空 |
| email | 邮箱 | Varchar(50) |  | 可空 |
| instit\_type | 类型 | Varchar(10) |  | 可空 |
| parent\_id | 父ID | Varchar(36) |  | 可空 |
| creator | 创建人 | Varchar(20) |  | 可空 |
| creationtime | 创建时间 | date |  | 可空 |
| ts |  | date |  | 可空 |
| dr |  | int |  | 可空 |

表格 2 联系人表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 名称 | 类型 | 主键 | 备注 |
| id | 主键 | Char(36) | 是 | Uuid |
| peocode | 编码 | Varchar(100) |  | 非空 |
| peoname | 姓名 | Varchar(50) |  | 非空 |
| sex | 性别 | Varchar(50) |  | 可空 |
| worktel | 工作电话 | Varchar(100) |  | 可空 |
| office | 办公室 | Varchar(100) |  | 可空 |
| tel | 电话 | Varchar(100) |  | 可空 |
| email | 邮箱 | Varchar(100) |  | 可空 |
| countryzone | 国家 | Varchar(36) |  | 可空 |
| institid | 组织id | Char(50) |  | 可空 |
| institname | 组织名称 | Char(50) |  | 可空 |

* 设计示例功能。
* 新增组织机构
* 修改组织机构
* 删除组织机构
* 新增人员信息
* 修改人员信息
* 删除人员信息
* 查询人员信息（根据某些字段做模糊查询）
* 树表联动查询（点击某个组织机构，查询其节点下所有人员以及其子孙节点下所有成员）

# 示例配置

## 数据库配置

在IUAP的数据库中创建

1、组织机构表“example\_organization”。

2、联系人表：example\_contacts

## 开发环境配置

用eclipse载入示例工程，选中contacts目录下的java和resources目录，然后右键—》Build Path—》Use as Source Folder，将print示例载入编译环境。

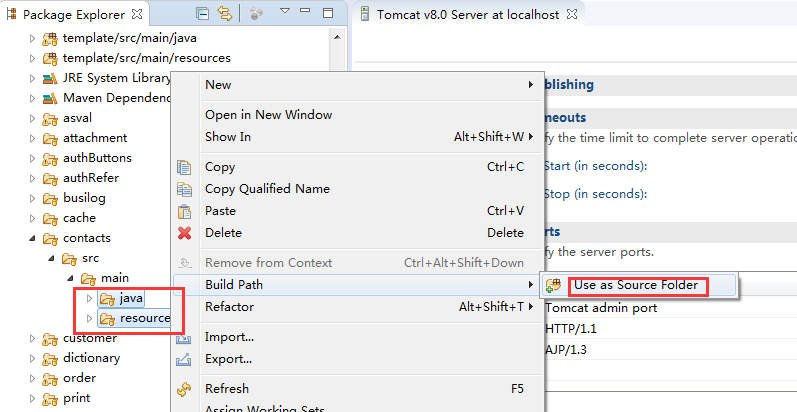


图 1 工程结构

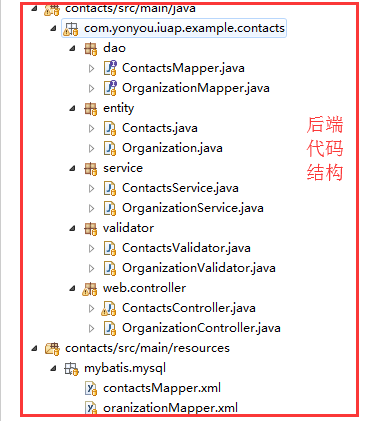


图 2 后端代码结构

前端代码结构

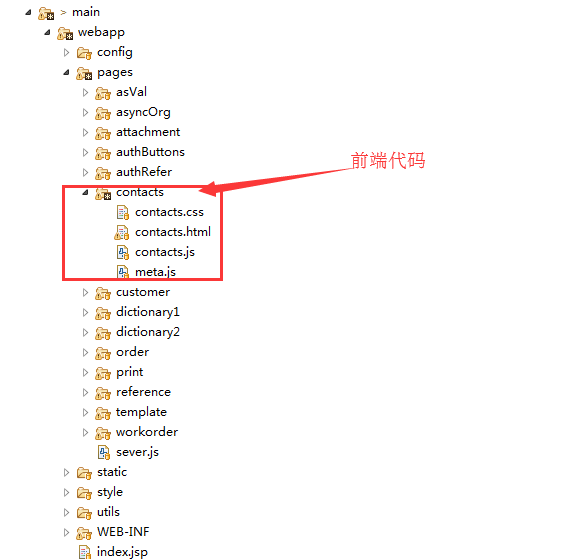


图 3 前端代码结构

检查项目无问题后，右击iuap\_demo –》Run As —》run on server。

## 功能测试

### 菜单注册

手工进行菜单注册，为标准方法，不重复介绍。

功能节点设置js地址为：/iuap-example/pages/contacts/contacts.js。

### 功能验证

1. 新增组织机构



图 4 新增组织机构

1. 修改组织机构信息

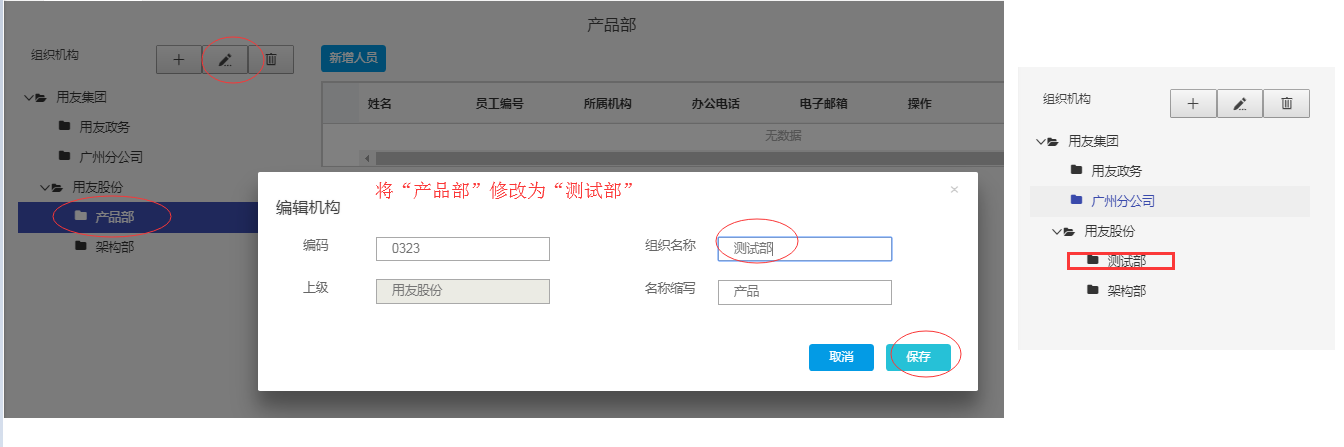


图 5 修改组织机构信息

1. 删除组织机构

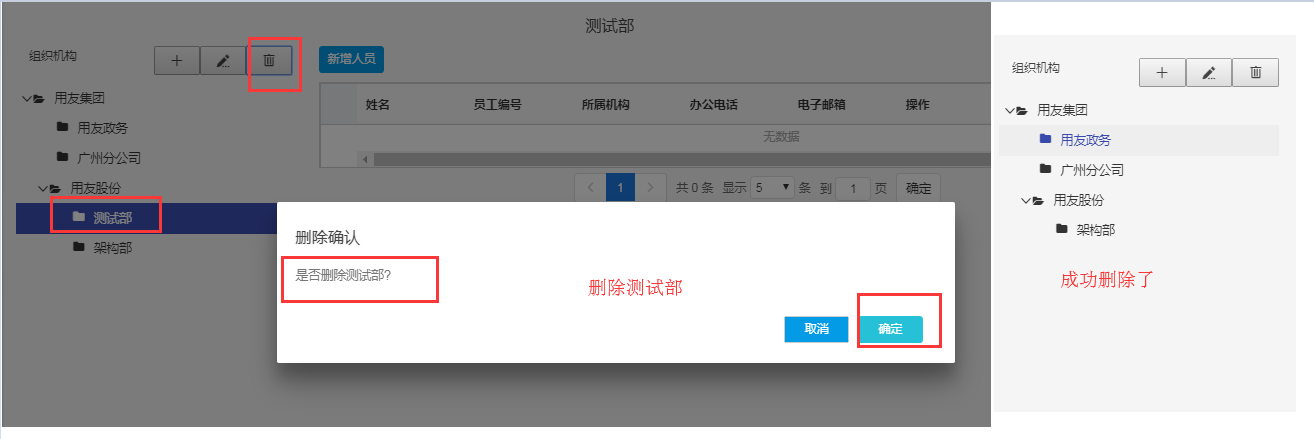


图 6 删除组织机构

1. 新增人员

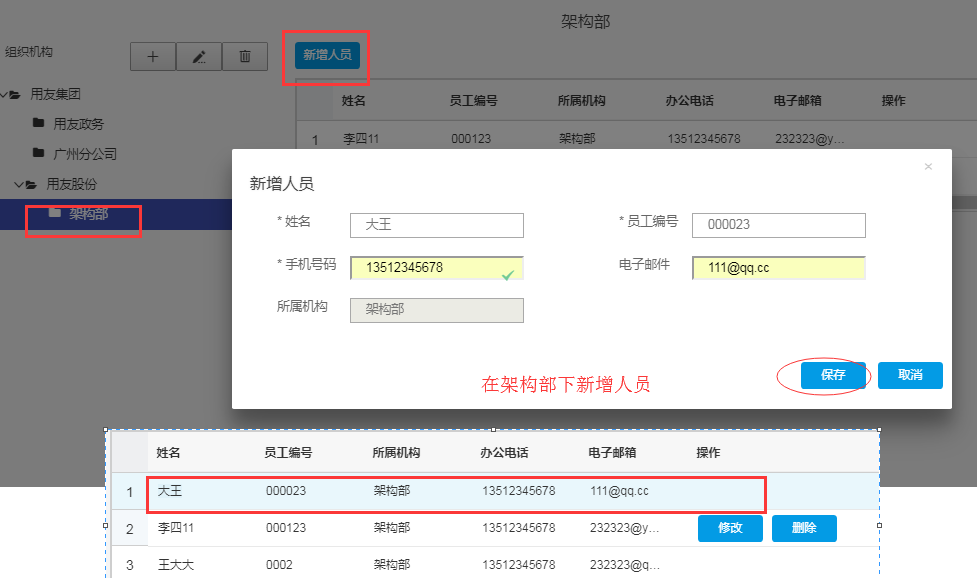


图 7 新增人员信息

1. 修改人员信息

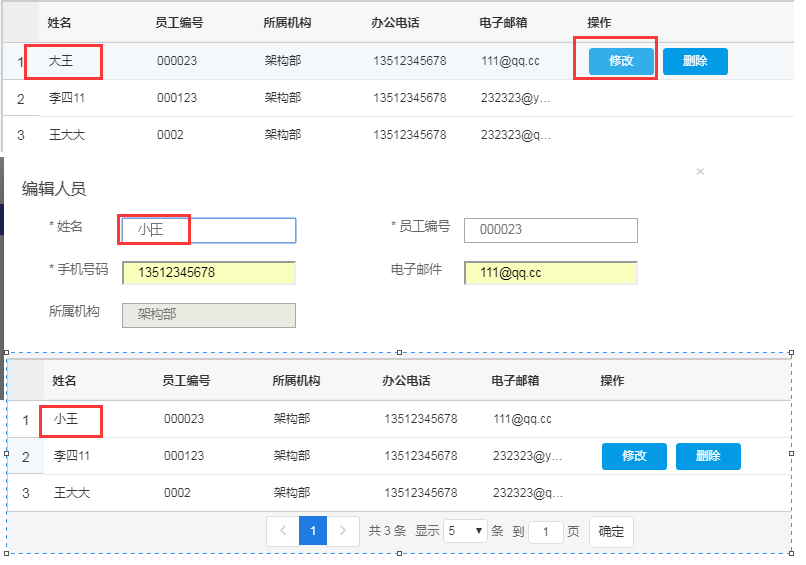


图 8 修改人员信息

1. 删除人员



图 9 删除人员信息

1. 查询人员



图 10 查询人员信息

1. 联动查询



图 11 联动查询

# 代码开发示例

## 后端开发

### 组织机构操作

1. 查询所有组织机构“/list”，主要是根据前端数据要求来组织结构。

@RequestMapping(value = "/list", method = RequestMethod.GET)

public @ResponseBody Object loadTree() {

List<Organization> data = oranizationService.findAll();

return buildSuccess(data);

}

1. 保存组织机构（包含新增和修改，通过主键判定）

@RequestMapping(value = "/save", method = RequestMethod.POST)

public @ResponseBody Object save(@RequestBody List<Organization> list) {

validtor.valid(list);

oranizationService.save(list);

return buildSuccess();

}

1. 删除组织机构（传入组织机构）

@RequestMapping(value = "/del", method = RequestMethod.POST)

public @ResponseBody Object del(@RequestBody List<Organization> list) {

oranizationService.batchDeleteByPrimaryKey(list);

return buildSuccess();

}

1. 根据父节点查询子节点（传入父Id）

@RequestMapping(value = "/findByFid", method = RequestMethod.GET)

public @ResponseBody Object findByFid(@RequestBody String pid) {

List<Organization> data = oranizationService.findByFid(pid);

return buildSuccess(data);

}

### 人员信息操作

1. 查询所有人员

需要提供分页显示，需要传入组织机构集合，需要传入查询条件。

@RequestMapping(value = "/list", method = RequestMethod.GET)

public @ResponseBody Object page(PageRequest pageRequest, SearchParams searchParams ,String institid) {

Page<Contacts> tmpdata = contactsService.selectAllByPage(pageRequest, searchParams,institid);

Page<Contacts> data = contactsService.selectInstit(tmpdata);

return buildSuccess(data);

}

1. 保存人员信息（包含新增和修改，通过主键判定）

@RequestMapping(value = "/save", method = RequestMethod.POST)

public @ResponseBody Object save(@RequestBody List<Contacts> list) {

validtor.valid(list);

contactsService.save(list);

return buildSuccess();

}

1. 删除人员信息

@RequestMapping(value = "/del", method = RequestMethod.POST)

public @ResponseBody Object del(@RequestBody List<Contacts> list) {

contactsService.batchDeleteByPrimaryKey(list);

return buildSuccess();

}

## 前端开发

### 编写meta.js

viewModel.js中定义数据模型，包含机构和人员的字段要求。

var meta = {

meta: {

institcode: {

type: "string",

required: true,

notipFlag: true,

hasSuccess: true,

nullMsg: "机构编码不能为空!"

},

institname: {

type: "string",

required: true,

notipFlag: true,

hasSuccess: true,

nullMsg: "机构名称不能为空!"

},

institid: {

type: "string"

},

parentid: {

type: "string"

},

parentname: {

type: "string"

},

shortname: {

type: "string"

}

}

};

var metaTelbook = {

meta: {

id: {

type: "string"

},

peoname: {

type: "string",

required: true,

notipFlag: true,

hasSuccess: true,

nullMsg: "姓名不能为空!"

},

peocode: {

type: "string",

required: true,

notipFlag: true,

hasSuccess: true,

nullMsg: "员工编号不能为空!"

},

institid: {

type: "string"

},

institname: {

type: "string"

},

worktel: {

type: "string",

regExp: /^1[34578]\d{9}$/,

notipFlag: true,

hasSuccess: true,

errorMsg: "手机号码格式不对。"

},

email: {

type: "string",

regExp: /^(\w-\*\.\*)+@(\w-?)+(\.\w{2,})+$/,

notipFlag: true,

hasSuccess: true,

errorMsg: "邮箱格式不对。"

},

tel: {

type: "string"

},

office: {

type: "string"

},

operate:{

type: "string"

}

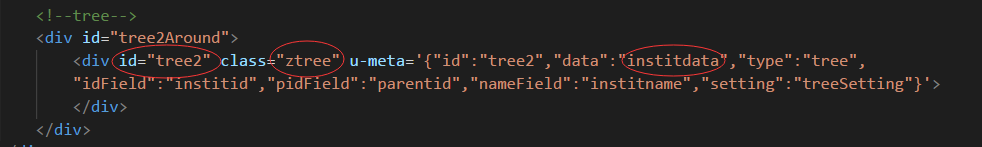
}

};

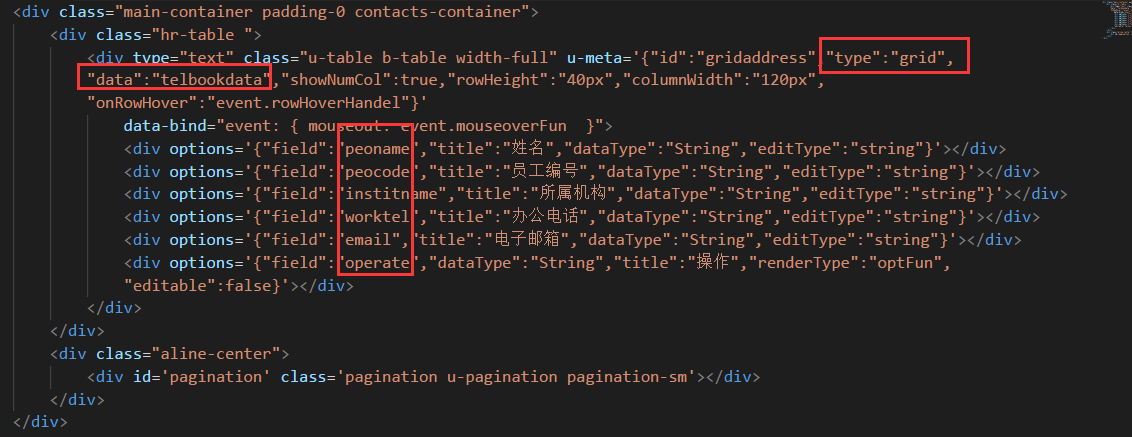
### 编写contacts.html文件

1. 左侧的树设置

设置的参数可以参看tinper的官方文档示例和api。



1. 右侧人员表的设置



### 编写contacts.css文件

文件中定义的样式可以根据实际情况修改。

### 编写contacts.js文件

* 添加对文件的引用，

define(

    ["text!./contacts.html", "css!./contacts.css", "./meta.js",

        "css!../../style/style.css", "../../config/sys\_const.js",

        "css!../../style/widget.css", "../sever.js",

        "css!../../style/common.css"],

* 修改调用服务的地址

var ctx = cookie.appCtx;

var treeListUrl = ctx + "/instit/list"; //机构查询

var treeDelUrl = ctx + "/instit/del/"; //机构删除

var treeSaveUrl = ctx + "/instit/save"; //机构新增和保存

var tableListUrl = ctx + "/telBook/list"; //人员查询

var tableDelUrl = ctx + "/telBook/del/"; //人员删除

var tableSaveUrl = ctx + "/telBook/save"; //人员新增和保存

* 注意Tree的设置，点击事件。

/\* 树设置 \*/

treeSetting: {

view: {

showLine: false,

selectedMulti: false

},

callback: {

onClick: function (e, id, node) {

treeid = [];

viewModel.event.getAllChildren(node, treeid);

var pid = node.pId;

viewModel.event.loadTelbook(treeid);

}

}

}