技 术 文 件

技术文件名称：表单设计器详细设计

技术文件编号：

版 本：V5.10.01

共 10 页

(包括封面)

拟 制

审 核

会 签

标准化

批 准

中兴通讯股份有限公司

修改记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 文件编号 | 版本号 | 拟制人/  修改人 | 拟制日期/  修改日期 | 更改理由 | 主要更改内容  （写要点即可） |
|  | 5.10.01 | 陈波 | 2015-06-01 | 新建 | 无 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 注：文件第一次归档时，“更改理由”、“主要更改内容”栏写“无”。 | | | | | |

目 录

[(包括封面) 1](#_Toc456797496)

[1 引言 5](#_Toc456797497)

[1.1 编写目的 5](#_Toc456797498)

[1.2 文档约定 5](#_Toc456797499)

[2 术语、定义和缩略语 5](#_Toc456797500)

[2.1 术语、定义 5](#_Toc456797501)

[2.2 缩略语 5](#_Toc456797502)

[3 使用技术 6](#_Toc456797503)

[3.1 Avalon 6](#_Toc456797504)

[3.2 Jquery 6](#_Toc456797505)

[3.3 Bootstrap 6](#_Toc456797506)

[4 使用说明 7](#_Toc456797507)

[4.1 框架说明 7](#_Toc456797508)

[4.1.1 组件面板 9](#_Toc456797509)

[4.1.2 布局容器 9](#_Toc456797510)

[4.1.3 属性面板 9](#_Toc456797511)

[4.1.4 工具栏 9](#_Toc456797512)

[4.2 整体流程 9](#_Toc456797513)

[4.2.1 Tomcat单独部署 9](#_Toc456797514)

[4.2.2 ICT部署 10](#_Toc456797515)

[4.2.3 表单设计 10](#_Toc456797516)

[4.2.4 表格设计 12](#_Toc456797517)

[4.3 框架提供接口 13](#_Toc456797518)

[4.3.1 表单页面URL参数说明 13](#_Toc456797519)

[4.3.2 URL参数获取getUrlParam(name, search) 13](#_Toc456797520)

[4.3.3 ORM支持查询条件格式 14](#_Toc456797521)

[4.3.4 组装条件说明 14](#_Toc456797522)

[4.3.5 表格接口 15](#_Toc456797523)

[4.3.6 对话框接口 20](#_Toc456797524)

[4.3.7 Map对象 21](#_Toc456797525)

[4.3.8 Ajax GET请求 21](#_Toc456797526)

[4.3.9 下拉框脚本数据加载 22](#_Toc456797527)

[4.3.10 批量插入接口 23](#_Toc456797528)

[4.3.11 Echarts图形接口 23](#_Toc456797529)

~~[4.3.12](#_Toc456797530)~~~~[路标导航接口](#_Toc456797530)~~ [24](#_Toc456797530)

[4.4 工程管理 25](#_Toc456797531)

[4.4.1 路径说明 25](#_Toc456797532)

[4.4.2 工程管理 25](#_Toc456797533)

[4.5 自定义数据源 27](#_Toc456797534)

[4.5.1 数据源信息 27](#_Toc456797535)

[4.5.2 数据服务信息 28](#_Toc456797536)

[4.5.3 数据集信息 28](#_Toc456797537)

[4.5.4 引用对象信息 30](#_Toc456797538)

[4.5.5 自定义json数据源示例 31](#_Toc456797539)

[4.6 数据集设置 31](#_Toc456797540)

[4.6.1 简介 31](#_Toc456797541)

[4.6.2 数据集列表 31](#_Toc456797542)

[4.6.3 新增、修改页面 31](#_Toc456797543)

[4.7 工具栏按钮 32](#_Toc456797544)

[4.7.1 新建 32](#_Toc456797545)

[4.7.2 打开 32](#_Toc456797546)

[4.7.3 保存 32](#_Toc456797547)

[4.7.4 导出 32](#_Toc456797548)

[4.7.5 预览 33](#_Toc456797549)

[4.7.6 工程管理 33](#_Toc456797550)

[4.7.7 备份 33](#_Toc456797551)

[4.7.8 脚本编辑器 33](#_Toc456797552)

[4.7.9 数据集设置 33](#_Toc456797553)

[4.7.10 使用手册 33](#_Toc456797554)

[4.8 组件使用 34](#_Toc456797555)

[4.8.1 全局属性 34](#_Toc456797556)

[4.8.2 布局 34](#_Toc456797557)

[4.8.3 标签 34](#_Toc456797558)

[4.8.4 文本框、文本域 34](#_Toc456797559)

[4.8.5 下拉框 35](#_Toc456797560)

[4.8.6 单选框 36](#_Toc456797561)

[4.8.7 日期 36](#_Toc456797562)

[4.8.8 上传 37](#_Toc456797563)

[4.8.9 按钮 37](#_Toc456797564)

[4.8.10 远程表格 38](#_Toc456797565)

[4.8.11 本地表格 40](#_Toc456797566)

[4.8.12 对话框 40](#_Toc456797567)

[4.8.13 脚本 41](#_Toc456797568)

[4.8.14 树 41](#_Toc456797569)

[4.8.15 分隔栏 42](#_Toc456797570)

[4.8.16 iframe 42](#_Toc456797571)

[4.8.17 Tab 42](#_Toc456797572)

[4.8.18 Collapse可折叠 42](#_Toc456797573)

[4.8.19 按钮组 42](#_Toc456797574)

[4.8.20 图形组件 43](#_Toc456797575)

~~[4.8.21](#_Toc456797576)~~~~[路标导航](#_Toc456797576)~~ [44](#_Toc456797576)

[4.8.22 移动端-购物车 44](#_Toc456797577)

[4.9 orm接口 47](#_Toc456797578)

[5 使用规范 48](#_Toc456797579)

[5.1 布局组件 48](#_Toc456797580)

[5.1.1 布局组件尽量嵌套使用，不要平铺使用。 48](#_Toc456797581)

[5.1.2 组件隐藏通过布局组件来实现，不能直接在样式中编写display:none 49](#_Toc456797582)

[6 问答 49](#_Toc456797583)

[6.1.1 表单设计中如何控制组件的灰化和使用？ 49](#_Toc456797584)

[6.1.2 为什么在表单查看时所有按钮都不见了？ 49](#_Toc456797585)

# 引言

## 编写目的

## 文档约定

# 术语、定义和缩略语

## 术语、定义

|  |  |
| --- | --- |
| 术语 | 解释 |
| UEP | 统一网管平台 |
| ICT |  |
| BPM |  |
| Activiti |  |

## 缩略语

|  |  |  |
| --- | --- | --- |
| **术语** | **英文释义** | **中文释义** |
| JVM | Java Virtual Machine | Java虚拟机 |
| J2EE | Java2 Platform Enterprise Edition | Java2平台企业板 |
| EJB | Enterprise JavaBean | 企业级JavaBean |
| JMS | Java Message Service | JAVA消息服务 |
| JS | JAVA SCRIPT | 网页脚本语言 |
| JSP | JAVA SERVER PAGE | JAVA服务端页面 |
| JDOM | Java Document Object Model | Java文档对象模型 |
| JCA | J2EE Connector Architecture | J2EE链接管理 |
| MML | Man Machine Language | 人机命令语言 |
| CLI | *Command Line Interface* | *命令行接口* |
| GUI | Graphic User Interface | 图形用户界面 |
| AAA(3A) | Authentication, Authorization and Accounting | 认证、授权和计费 |
| AAAA(4A) | Authentication, Authorization, Accounting and Audit | 集中认证、集中授权、集中帐号和集中审计 |
| SSO | Single Sign-On | 单点登录 |

# 使用技术

表单设计器使用了jquery、avalon框架。界面风格使用bootstrap3.0。

## Avalon

Avalon（http://avalonjs.github.io/）司徒正美开发和维护的一个简单易用迷你的MVVM框架，它最早发布于2012.09.15，为解决同一业务逻辑存在各种视图呈现而开发出来的。MVVM将所有前端代码彻底分成两部分，可以轻松实现数据的隔离和双向绑定，视图的处理通过绑定实现（angular有个更炫酷的名词叫指令）， 业务逻辑则集中在一个个叫VM的对象中处理。我们只要操作VM的数据，它就自然而然地神奇地同步到视图。

参考资料：

[http://avalonjs.github.io/#zh/apis/index.html](http://avalonjs.github.io/" \l "zh/apis/index.html)

<http://www.html-js.com/article/column/234>

## Jquery

jQuery，顾名思义，就是JavaScript和查询（Query），即是辅助JavaScript开发的库。它的核心理念是write less,do more(写得更少,做得更多)。

Jquery是继prototype之后一个优秀的Javascript库。它是轻量级的javascript库 ，它兼容CSS3，还兼容各种浏览器（如：IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+），jQuery2.0及后续版本将不再支持IE6/7/8浏览器，目前已经发布了jQuery3.0。

jQuery能够使用户的HTML页面保持代码和HTML内容分离。也就是说，不用再在HTML里面插入一堆js来调用命令了，只需要定义id即可。

jQuery使用户能更方便地处理HTML、events、实现动画效果，并且方便地为网站提供AJAX交互。

jQuery的语法设计可以使开发者更加便捷，例如操作文档对象、选择DOM元素、制作动画效果、事件处理、使用Ajax以及其他功能。

除此以外，jQuery提供API让开发者编写插件。其模块化的使用方式使开发者可以很轻松的开发出功能强大的静态或动态网页。

jQuery的文档说明很全，而且还有许多成熟的插件可供选择。jQuery是免费、开源的，使用MIT许可协议。

现在，jQuery已经成为最流行的javascript库，在世界前10000个访问最多的网站中，有超过55%在使用jQuery。

## Bootstrap

Bootstrap是基于HTML5和CSS3开发的，它在jQuery的基础上进行了更为个性化和人性化的完善，形成一套自己独有的网站风格，并兼容大部分jQuery插件。

组件

Bootstrap中包含了丰富的Web组件，根据这些组件，可以快速的搭建一个漂亮、功能完备的网站。其中包括以下组件：下拉菜单、按钮组、按钮下拉菜单、导航、导航条、面包屑、分页、排版、缩略图、警告对话框、进度条、媒体对象等。

Javascript插件编辑

Bootstrap自带了13个jQuery插件，这些插件为Bootstrap中的组件赋予了“生命”。其中包括：模式对话框、标签页、滚动条、弹出框等。

框架代码编辑

可以对Bootstrap中所有的CSS变量进行修改，依据自己的需求裁剪代码。



参考资料：

<http://www.bootcss.com/>

# 使用说明

## 框架说明

表单设计器框架结构如下图所示

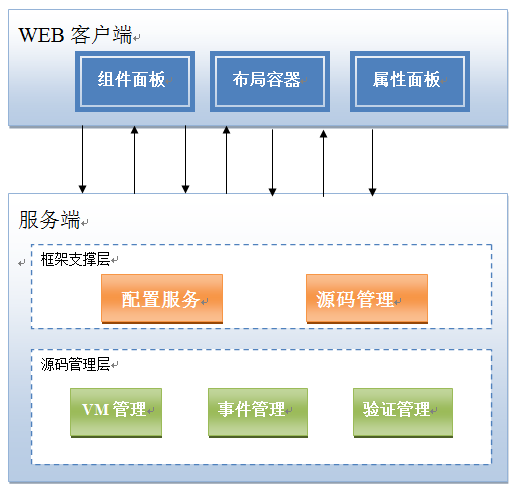
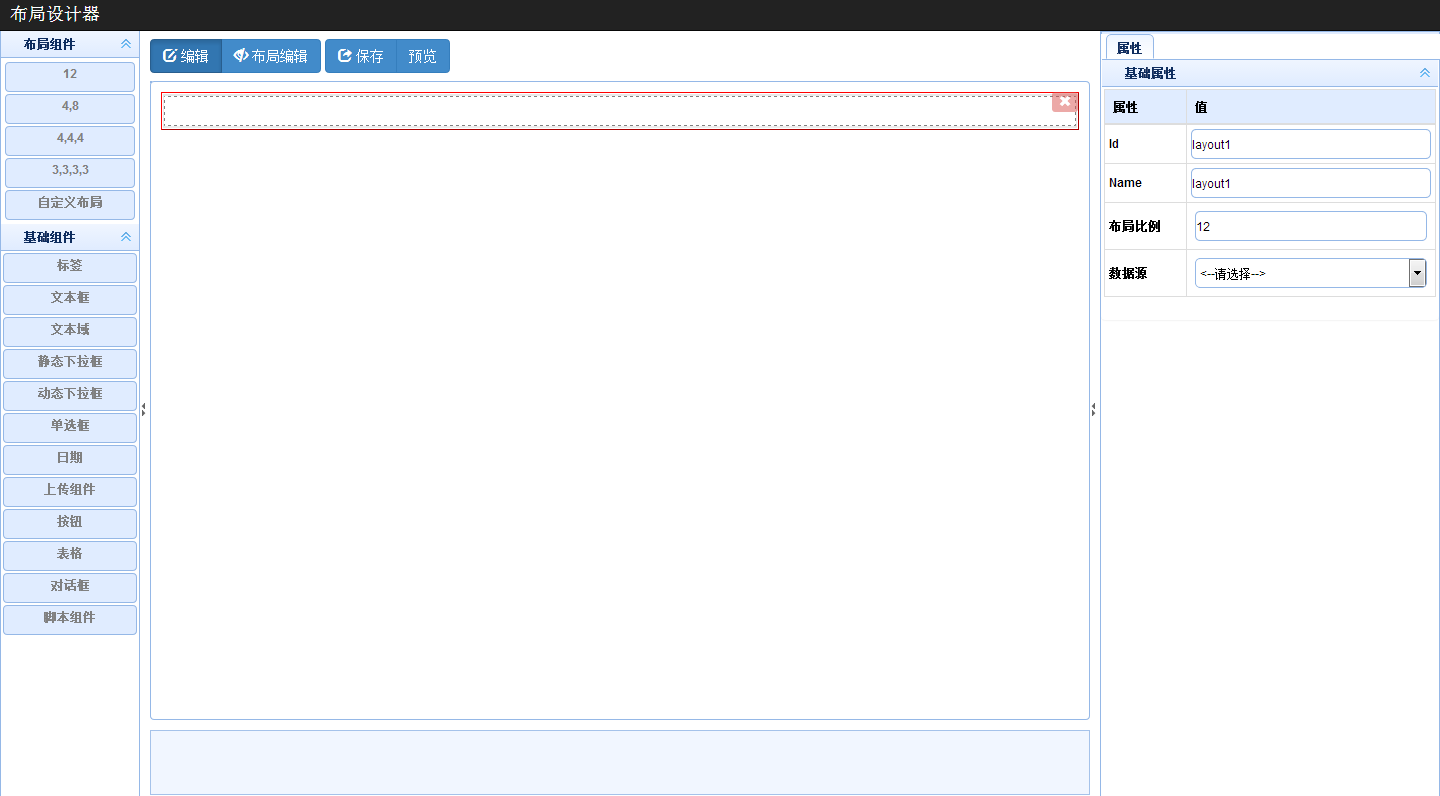


图5.1 表单设计器框架

设计界面：



### 组件面板

组件面板如上图左边蓝色区域部分，按其用途分为布局组件和基础组件，后续根据业务扩展会增加功能组件和业务组件等其他分类。布局组件支持直接拖拉到布局容器中，其余组件只能拖拉到布局组件上。

### 布局容器

布局容器分为设计器容器和隐藏组件容器，设计器容器如上图中间白色区域部分，该区域可根据业务设计表单和列表页面；隐藏组件容器如上图中间蓝色区域部分，该区域只支持拖拉页面隐藏组件，现在该类组件只有对话框和脚本两类组件。

### 属性面板

属性面板如上图左边区域部分，该部分提供了所有组件的属性编辑，该面板提供基础属性、高级属性和事件三个tab页配置属性，后续根据业务要求调整tab页内容和tab页数量。所有属性编辑的内容都放置到被设置组件的html标签中，待需要使用时从标签中读出需要的属性供其使用。

### 工具栏

新建：新增框架设计文件。

打开：打开设计文件。

保存：保存当前设计文件。

预览：预览时需要先保存设计文件，这样预览页与设计页面才能同步；如果工程本地路径与发布路径不同时，表单设计器会进行一次发布操作，然后才会弹出预览页面。

工程管理：包含工程管理、数据源管理和设计文件管理。工程管理用于配置工程相关信息包括：工程名、工程源码保存路径、工程发布路径、用户自定义JS相对路径和预览端口。数据源管理用于配置业务数据源接口信息，包括：数据源ID、数据源名称、IP、端口、URL前缀等。文件管理用于配置工程设计文件信息，包括文件相对于工程所在的目录、文件名称。

备份：备份当前的设计文件和源文件。

详细说明参见：[工具栏按钮章节](#_工具栏按钮)

## 整体流程

### Tomcat单独部署

ORM

ORM生成的是war工程，部署时把war里的内容解压到dataservice目录并部署到tomcat中。修改dataservice/WEB-INF/classes/ db.properties文件，各参数说明在该文件中均有描述。

注意：如果不使用ORM提供的服务可不部署ORM。

表单设计器

表单设计器生成的是war工程，部署时把war里的内容解压到designer目录并部署到tomcat中。修改designer/WEB-INF/classes/ form-designer-project.xml文件。该文件内容可直接修改或者在表单设计器页面中调整，标签说明都放到xml文件中。

启动tomcat，访问<http://ip:port/designer/index.html>页面查看表单设计器是否部署成功。

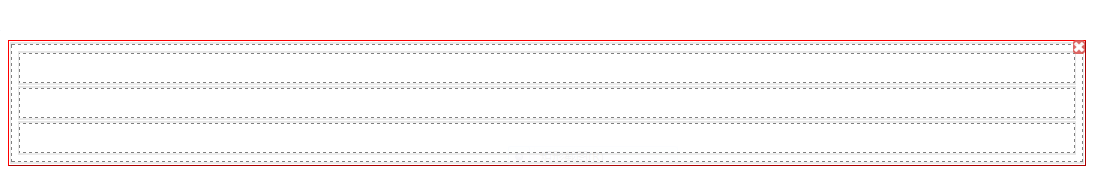
### ICT部署

从jenkins服务器上找到需要的版本，下面红色部分内容BCP-BFB-\*.ZIP(表单设计器) BCP-DSS-\*.ZIP(ORM),解压后直接覆盖到已安装好的ICT环境中。重启ICT后登陆ICT，访问http://:ip:port/dataservice/orm/metadata/tablenames是否有数据返回验证orm是否部署成功；访问<http://ip:port/designer/index.html>查看设计器页面验证表单设计器是否部署成功。



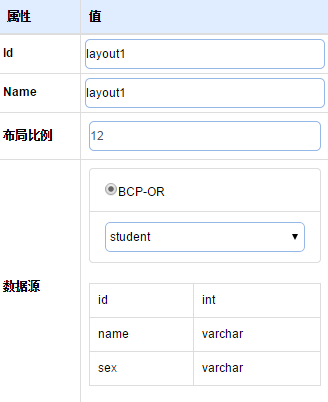
### 表单设计

如需使用表单设计器提供的新增、修改、删除功能则需要在顶层布局组件上选择数据源，顶层内的所有布局器的数据源都继承于该布局组件。

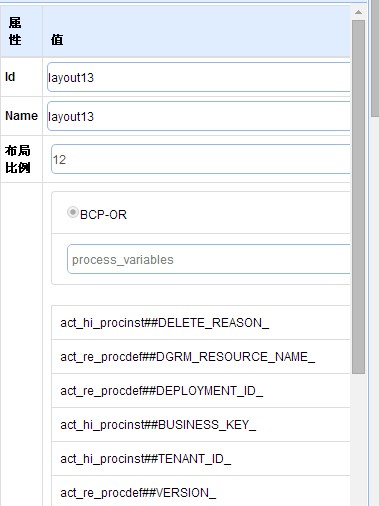


布局组件和顶层布局组件图

数据源分为单表和多表，单表可用于表单设计做新增、修改、删除等功能。多表只能做表格查询和显示功能。从数据源名称中无法看出是单表还是多表数据源，只有在选择数据源后显示数据列时显示有区别，多表数据列中都有“##”符号。

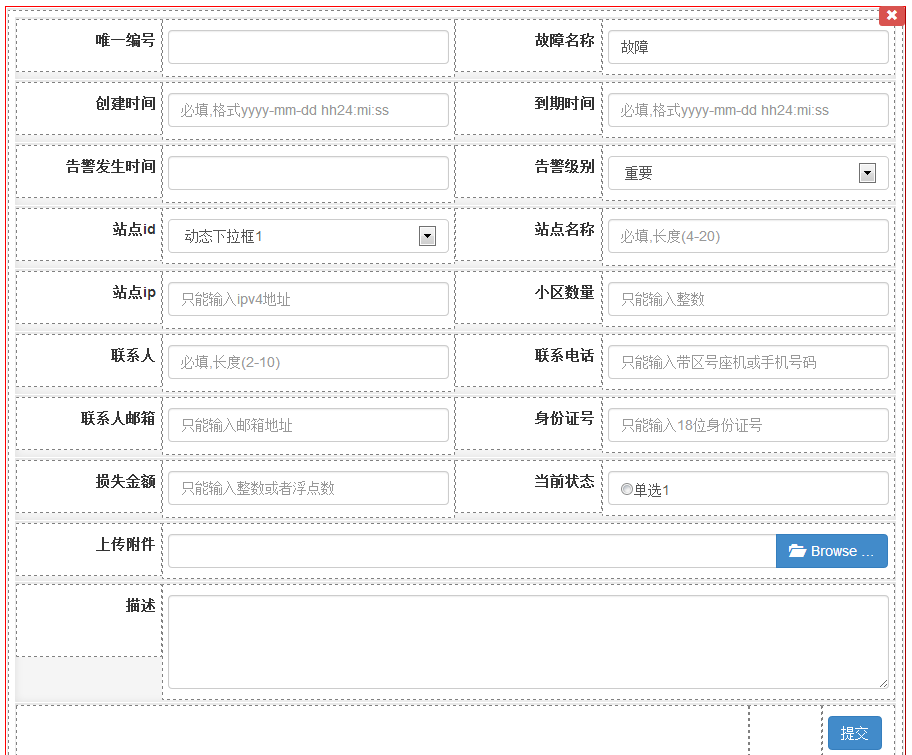


单表数据源选择图



多表数据源选择图

设计阶段表单设计器界面如图：

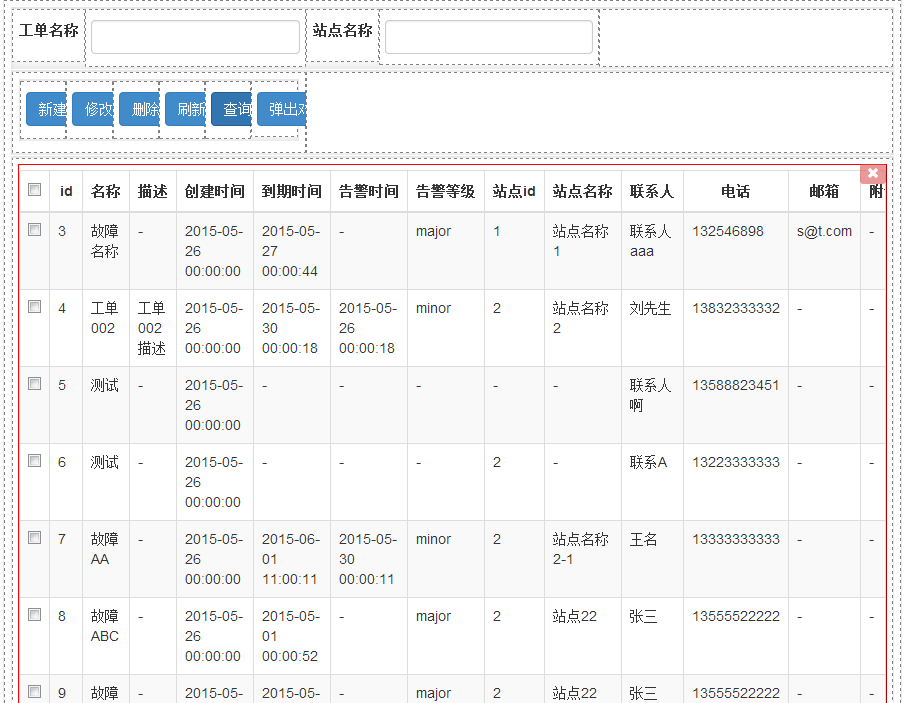


设计步骤：

1. 用户拉动一个布局比例为12的布局组件放到布局容器中。
2. 为该布局器设置数据源，该数据源只能是单表数据源，并包含设计表单的所有字段信息，在布局组件中加入其他组件后数据源不能修改。
3. 根据页面美观和设计要求拉动布局组件到已有的布局组件中，建议里层布局器都设置为2,4,2,4风格，效果和上图一样，布局组件支持嵌套。
4. 拉动需要的基础组件到布局组件中，为每个输入、选择和勾选的组件绑定数据字段，设计每个组件的属性信息，并在表单页面加入提交按钮。
5. 点击保存按钮会保存设计器框架代码和源文件，源文件包含html和js两个文件。

### 表格设计

设计阶段表单设计器界面如图：



设计步骤：

1. 用户拉动一个布局比例为12的布局组件放到布局容器中。
2. 为该布局器设置数据源，该数据源表格所有字段信息，在布局组件中加入其他组件后数据源不能修改。
3. 拉动表格组件到布局器中，设置表格基础属性、高级属性和事件等信息，点击应用按钮生成表格。
4. 点击保存按钮会保存设计器框架代码和生成页面代码，页面代码包含html和js两部分代码。

## 框架提供接口

### 表单页面URL参数说明

表单设计时会有“**可编辑性**”选项，该选项是通过表单url中的参数operator进行判断的。新增表单时需要传入operator=add、修改时传入operator=edit、查看时传入operaor=view，查询界面不需要传入参数。

如果表单设计时参数传入有误表单功能会不正确。

### URL参数获取getUrlParam(name, search)

|  |
| --- |
| /\*\*  \* 根据参数名获取url参数  \* @param name 参数名  \* @param search url所有参数  \* 如果是调用showModalDialog("dialog1", "新增表单","chenbo-form.html?operator=add")打开页面，需要获取该url中参数，search使用pageParams（该对象为框架定义的全局变量）  \* 如果是在本页面获取参数，使用window.location.search。  \* @returns 参数值，如果为空则返回null。\*/  getUrlParam(name, search); |

### ORM支持查询条件格式

|  |
| --- |
| 多个条件使用：  "condition" : {  "and" : [ {"cname" : "a","value" : 1,"compare" : "="  }, {  "or" : [ {"cname" : "b","value" : 1, "compare" : "="  }, {  "cname" : "c","value" : 1, "compare" : "="  } ]  } ]  }  单个条件使用：  "condition": {"cname" : "id","value" : 1,"compare" : "="} |

### 组装条件说明

一层条件组织

例如：where a=1 and b=2 and c=3这种形式。

|  |
| --- |
| /\*\*  \* 根据查询条件数组组装条件  \* @param conditions 条件数组  \* eg:  var conditions = [];  var condition = new QueryCondition();  condition.setCName("columnname");  condition.setValue("columnvalue");  condition.setCompare("=");  conditions.push(condition); // 多个条件push多个  \* @param operator 只能输入and/or字符串  \* @returns 条件对象  \*/  function generateCondition(conditions, operator) |

例如 where c1=1 and (c2=2 or c3=1)这种组织形式

|  |
| --- |
| var c1 = new QueryCondition();  c1.setCName("c1").setCompare("=").setValue("1");  var c2 = new new QueryCondition();  c2.setCName("c2".setCompare("=").setValue("2");  var c3 = new QueryCondition();  c3.setCName("c3").setCompare("=").setValue("1");  var c23 = generateCondition([c2,c3],"or");  var call = generateCondition([c1,c23],"and"); |

### 表格接口

源码路径: js/page/layoutitPageTable.js和js/lib/bootstrap-table-editable.js

#### 获取表格选中行接口(远程/本地表格通用)

/\*\*

\* 获取表格选中数据，需要传入表格id参数，返回为数组

\* @param id 表格id

\*/

function getTableSelectData(id)

#### 获取表格所有行接口(远程/本地表格通用)

/\*\*

\* 获取表格所有数据，需要传入表格id参数，返回为数组

\* @param id 表格id

\*/

function getTalbeAllData(id)

#### 表格刷新(远程/本地表格通用)

/\*\*

\* 刷新表格数据，如果不传入options则只做刷新

\* @param id 表格id

\* @param options 表格支持的属性

\*/

function refreshTable(id,options)

#### 获取表格属性(远程/本地表格通用)

|  |
| --- |
| /\*\*  \* 根据表格id和属性名称获取表格属性  \* @param id 表格id  \* @param option 表格选项，  \* 例如： columns包含所有字段，拿到数据后需要过滤掉不需要字段。  \* url为具体url  \*/  function getTableOptions(id, option) |

#### 远程表格查询接口

代码位置：js/page/ layoutitPageTable.js

|  |
| --- |
| /\*\*  \* 表格查询操作  \* @param id 表格id  \* @param conditions 过滤条件  \* 单个条件设置方法：  \* var condition = new QueryCondition();  \* condition.setCName("columnname");  \* condition.setValue("columnvalue");  \* condition.setCompare("=");  注意需要把:需要把条件放到数组中  \*/  function queryRemoteTable( id, conditions) |

#### 远程表格删除接口

|  |
| --- |
| /\*\*  \* 删除远程表格数据  \* @param id 表格id  \* @param dsname 数据源名称  \* @param condition 删除条件，该条件为组织好的查询条件，单层条件可以使用generateCondition方法组织，嵌套条件  \* @param tablename 需要删除数据的数据表表名，只有在表格绑定的数据是多表时才需要传入这个参数，条件中的字段名必须保持与该表字段一致  \* @param successCallback 删除成功后的回调函数，回调方法有data参数，data包含两个属性{status: 1, message: "success"} ，其中status为1表示成功，其余情况为失败，message在status不为1时返回服务端的错误信息。  \*/  function deleteRemoteTableData(id, condition, tablename, successCallback) |

#### 批量更新远程表格数据接口

|  |
| --- |
| /\*\*  \*更新远程表格多行数据，行数据直接在表格中修改  \* @param tableName需要更新的表名，只有在更新的表与表格绑定的表不一致是才需要传入该参数 eg：表格绑定的是多表数据，这个时候是无法直接更新表数据的  \* @param validate验证方法，如果不需要验证则传入undefined  \* @param success成功后的回调，会把服务端返回结果作为参数传入到该方法中  \* @example $().bootstrapTable("updateRemoteTableRows",{  \* tableName:"t\_xt\_user", //optional  \* validate:function({tableId:"local",data:data}){  \* return {  \* message:"",  \* status:true/false  \* }  \* }, //optional  \* success:function(){} //optional  \* })  \*/ |

#### 远程表格根据URL参数过滤数据

需要该功能时需要在js脚本定义变量isFilterTableData = true，否则功能不生效。

在一个页面有多个表格或者一个页面既有表单数据又有表格数据时建议不要开启该功能，否则数据显示不正确。

#### 远程表格初始化前调整表格参数接口

该js方法需要用户编写，方法名为tableid+” TableParamConfig”，参数为bootstraptable的参数，这里给出初始化表格时需要在url中加条件的示例

|  |
| --- |
| /\*\*  \* 过滤表格id为table\_base1439188259868数据，默认表格无查询条件，通过下面方法新增查询条件currentNodeId=servicetask1。  \* 表格需要查询的列和条件都是放到url中的，加入查询条件其实是更新url，表格查询url形式为：  http://10.74.216.19:8080/dataservice/orm/table/prule?param=%7B%22columns%22%3A%5B%7B%22cname%22%3A%22currentNodeId%22%7D%2C%7B%22cname%22%3A%22currentNodeName%22%7D%2C%7B%22cname%22%3A%22currentNodeType%22%7D%2C%7B%22cname%22%3A%22jumpToNodeId%22%7D%2C%7B%22cname%22%3A%22jumptoNodeName%22%7D%2C%7B%22cname%22%3A%22nextNodeId%22%7D%2C%7B%22cname%22%3A%22nextNodeName%22%7D%2C%7B%22cname%22%3A%22note%22%7D%2C%7B%22cname%22%3A%22processDefId%22%7D%2C%7B%22cname%22%3A%22processDefName%22%7D%2C%7B%22cname%22%3A%22ruleName%22%7D%2C%7B%22cname%22%3A%22ruleOrder%22%7D%2C%7B%22cname%22%3A%22ruleScript%22%7D%2C%7B%22cname%22%3A%22serviceType%22%7D%2C%7B%22cname%22%3A%22serviceURL%22%7D%2C%7B%22cname%22%3A%22variableName%22%7D%5D%2C%22orders%22%3A%5B%5D%2C%22condition%22%3A%7B%22cname%22%3A%22currentNodeId%22%2C%22compare%22%3A%22%3D%22%2C%22value%22%3A%22servicetask1%22%7D%7D  \*  \*/  function table\_base1439188259868TableParamConfig(param) {  var url = param.url;  var obj = JSON.parse(decodeURIComponent(url.substring(url.indexOf("param=")+6)));  var conditions = [];  //如果有多个条件，需要new多个querycondition并push到conditions中。  var condition = new QueryCondition();  condition.setCName("currentNodeId").setCompare("=").setValue("servicetask1");  conditions.push(condition);  obj.condition = generateCondition(conditions, "and");    url = url.substring(0, url.indexOf("param=")) + "param=" + encodeURIComponent(JSON.stringify(obj));  param.url = url;  //这里必须renturn  return param;  } |

#### 表格添加数据

|  |
| --- |
| /\*\*  \* 加载到当前数据的末尾  \* @param data Array  \* @description 数据的可编辑性能生效的前提是数据对应字段具有可编辑性  \* 字段属性如下:  \* columns:[  \* {  \* field:"age",  \* editable:true,  \* validate:"custom"  \* },  \* {  \* field:"ete",  \* editable:false,  \* validate:''  \* }  \* ]  \* @example $().bootstrapTable("appendData",  \* [  \* {  \* age:{  \* value:111,  \* editable:false  \* }  \* },  \* {  \* aeta:14,  \* age:{value:14,editable:true},  \* ete:15,  \* test:{value:14,editable:false}  \* }  \* ]  \* )  \*/ |

#### 表格设置列是否可编辑

updateTableEditable

|  |
| --- |
| /\*\*  \* 表格可编辑性设置  \* @param params{rowIndex,fieldName,editable}  \* @example  \* $("#tableId").bootstrapTable("updateTableEditable",{  \* rowIndex:1,  \* fieldName:'name',  \* editable:true  \* })  \*/ |

#### 本地表格单元格编辑完成后回调接口

该接口由应用模块实现，完成在表格数据变化后的相关操作。

|  |
| --- |
| /\*\*  \* 本地表格列数据编辑后的回调函数  \* @param tableid 表格Id  \* @param index 行号  \* @param field 修改单元格field  \* @param row 修改行记录  \*/  function updateLocalTableCallback(tableid,index, field, row) |

#### 表格显示指定列。

BootstrapTable扩展方法*showTableColumns*支持隐藏指定列

|  |
| --- |
| /\*\*  \* @description 显示指定列  \* @param field {}/[]  \* @example $("#tableId").bootstrapTable("showTableColumns","name");  \* @example $("#tableId").bootstrapTable("showTableColumns","operator"),  \* $("#tableId").bootstrapTable("showTableColumns",["name","age","sex"]);  \*/ |

#### 表格隐藏指定列。

BootstrapTable扩展方法*hideTableColumns*支持隐藏指定列

|  |
| --- |
| /\*\*  \* @description 隐藏指定列  \* @param field {}/[]  \* @example $("#tableId").bootstrapTable("hideTableColumns","name");  \* $("#tableId").bootstrapTable("hideTableColumns",["name","age","sex"]);  \*/ |

#### 表格自定义高级搜索框。

该功能点框架提供两个符合规则的方法扩展。

1、返回自定义搜索组件的html

方法命名规则："表格ID"+"\_search"

|  |
| --- |
| function table12345\_search(){  return "<div></div>";  } |

2、自定义搜索按钮事件

方法命名规则："表格ID"+"\_search"

|  |
| --- |
| function table12345\_searchEvent(){  /\*\*  code  \*/  } |
|  |

两个自定义方法必须同时实现

#### 添加表单数据到本地表格接口

此函数用于将表单数据添加到本地表格中，表单数据来源于调用该函数的宿主对象所在的表单区域。

|  |
| --- |
| /\*\*  \* 添加表单数据到本地表格  \* @param tableId 本地表格id  \*/  function addFormDataToLocalTable(tableId) |

### 对话框接口

源码路径: js/page/layoutitPageDialog.js

#### 对话框引用页面注意事项

对话框加载的文件中，JQuery方法$(document).ready()不会等该页面加载完后才执行，而会顺序执行，在开发对话框加载页面时不要把该页面相关数据定义到其他公共JS文件中，不然会导致数据无法加载的情况。

#### 弹出对话框

uri只能输入当前页面访问路径的相对路径，支持“../”往上一层页面的写法，**uri中不能输入“/test.html”这种返回到根路径写法。**

|  |
| --- |
| /\*\*  \* 提供模式对话框接口，  \* @example showModalDialog("dialogid") 无标题静态对话框，对话框内容在页面中已定义。  \* showModalDialog("dialogid",title) 有标题静态对话框，对话框内容在页面中已定义。  \* showModalDialog("dialogid",title, uri) 有标题动态对话框，对话框内容由传入的url确定。  \* @param id 对话框id  \* @param title 对话框标题  \* @param url 对话框url，url为父页面的相对路径  \*  \*/  function showModalDialog(params) |

注意：对话框URL中传入的参数只能通过全局变量pageParams获取，不能直接获取地址栏参数。

#### 关闭对话框

/\*\*

\* 自定义手动关闭对话框

\* @author freedom

\* @param id 对话框id

\*/

function hideModalDialog(id)

### Map对象

使用方式同java类似。

var map = new Map();

map.put(key, value);

map.get(key);

map.size();

### Ajax GET请求

/\*\*

\* 根据参数发送ajax请求，只需要定义相应参数即可

\* @param ajaxParameter AjaxParameter对象实例，包括url data async和 callBack调用。

\* eg:var param = new AjaxParameter(); param.url = ....设置需要参数

\* ajaxgetService(param)

\*/

function ajaxgetService(ajaxParameter)

### 下拉框脚本数据加载

/\*\*

\* 根据表名，显示名、值等信息加载下拉框数据。<br />

\* eg: var zrdepartOptions=[]; getSelectOption("zrdepart","departname","id", zrdepartOptions, condition, isDistinct);

\* @param table 需要查询表名

\* @param name 显示名称列

\* @param value 显示值列

\* @param options 需要加载的数组对象，该对象必须外部定义。

\* @param condition 过滤条件，如果没有条件可以不传入该对象或者传为"undefined"

\* @param isDistinct 是否过滤掉重复数据，true表示需要过滤，false表示不需要过滤，不传值默认为false

\*/

\* @customVal 用户自定义的值，此值会添加在select的最前面,没有可以传为"undefined"。(此值为一个对象或者是一个对象数组-对象属性为name和value，形式{“name”:” ”,“value”:” ”})传入

\*/

function getSelectOption(dsname,table, name, value, options, condition, isDistinct,customVal )

#### 下拉框联动数据加载

源码路径：js/page/layoutitPage.js-->SelectAssociate方法

|  |
| --- |
| /\*\*  \* 不使用ORM公用的加载联动下拉框的选项  \* @param sourceIds 源下拉框ID数组 可以是多个下拉框的表单设计器生成的ID  \* @param targetId 联动下拉框ID 产生联动效果的下拉框的ID  \* @param resultData 传入的联动下拉框的选项，需要按照严格的下拉框的数据格式传入。  \*/  function selectValuesForApp(sourceIds, targetId, resultData)  /\*\*  \* 使用ORM方式获取，并且加载联动下拉框的选项  \* @param sourceId 源下拉框ID数组  \* @param targetId 联动下拉框ID  \* @param getURLCallBack 使用者定义的回调函数返回ORM的URL  \*/  function selectValuesForOrm(sourceIds, targetId, getURLCallBack)  以ORM方式的调用方法为例，具体的示例：  var idArr = [];  idArr.push("select\_dynamic1448329666785");  idArr.push("select\_dynamic1a448329668407");  var selectAssociate = new SelectAssociate();  selectAssociate.selectValuesForOrm(idArr,"select\_dynamic1448329670056",callback);  其中callback即使用者自定义的一个函数，返回内容为ORM的URL，URL的格式按照下拉框数据加载的形式返回，以如下为例：  function callback(conditionMap){  var condition1 = conditionMap.get("select\_dynamic1448329666785")[0];  var condition2 = conditionMap.get("select\_dynamic1448329668407")[0];  var url = "http://xxxx:xxxx/dataservice/orm/table/bcp\_re\_orgpart?param=  {\"columns\":[{\"cname\":\"partName\"},{\"cname\":\"partId\"}],\  "condition\":{\"and\":[{\"cname\":\"catelogId\",\"value\":"+condition1+",\"compare\":\"=\"},  {\"cname\":\"level\",\"value\":"+condition2+",\"compare\":\"=\"}]}}";  return url;  } |

### 批量插入接口

源码路径：js/page/layoutitPage.js-->batchInsert方法

|  |
| --- |
| /\*\*  \* 批量插入操作  \* eg：  \* var datas = [];  \* datas.push({processDefId:'processDefId\_value',processDefName:'processDefName\_value'});  \* datas.push({processDefId:'processDefId\_value2',processDefName:'processDefName\_value2'});  \* batchInsert("bcp", "prule", datas, function(data) {console.log(data)}); //bcp为dsname，prule为表名  \* @param dsname 数据源名称  \* @param tablename 表名  \* @param datas 数据数组，数组里的一个对象表示一调记录，只传入需要插入列的名称和值，形式为[{columnname1:value1,columnname2:value2,…},…]  \* @param successCallback 插入成功后的回调函数，回调方法有data参数，data包含两个属性{status: 1, message: "success"} ，其中status为1表示成功，其余情况为失败，message在status不为1时返回服务端的错误信息。  \*/  function batchInsert(dsname, tablename, datas, successCallback) |

### Echarts图形接口

#### 更新Echarts图标数据

该方法分为append(在现有数据基础上增加，使用与实时更新，折线图/柱状图/饼图)

，load(重新加载数据)

|  |
| --- |
| /\*\*  \* 更新ECharts图表数据  \* @param chart echarts对象 required  \* @param method 方法名称 load/append required  \* @param series 系列数据 [[seriesIdx,data,isHeaddataGrow,additionData],[seriesIdx,data]]  \* @example updateEChartSeriesData("id","load",[ //饼图重载数据  \* {  \* seriesIdx:0,  \* data:[  \* {value:335, name:'直接访问'}, \* {value:310, name:'邮件营销'}, \* {value:234, name:'联盟广告'}, \* {value:135, name:'视频广告'}, \* {value:1548, name:'搜索引擎'} \* ] \* } \* ]) \* @example updateEChartSeriesData("id","load",[ //折线图、柱状图重载数据 \* { \* seriesIdx:0, \* data:[ \* 335,120,300,546,213,145,2215,212,125,212,122,35 \* ] \* } \* ])  \* @example updateEChartSeriesData("id","append",[ //饼图添加数据 \* { \* seriesIdx:0, \* data:{name: '随机数据2',value:Math.round(Math.random()\*1000)}, \* additionData:'随机数据2' \* } \* ]) \* @example updateEChartSeriesData("id","append",[ //折线图、柱状图添加数据 \* { \* seriesIdx:0, \* data:25, \* additionData:'2015-08-26' \* } \* ]) \*/ |

### ~~路标导航接口~~

~~接口文件位于\js\page\layoutitWaypoint.js中。示例说明如下：~~

|  |
| --- |
| ~~//路标组件容器对象~~  ~~var object = $(“#waypoint”);~~  ~~//组件节点数据~~  ~~var data = [{"targetId":"layout1447759614814","name":"step1","icon":"","status":"0","initStatusIcon":"clear.png","saveStatusIcon":"clear-gray.png"}]~~  ~~//新建实例~~  ~~var waypoint = new LayoutitWaypoint();~~  ~~//初始化组件~~  ~~waypoint.init(object, data);~~  ~~//更新组件保存状态~~  ~~waypoint.updateNodeStatus(targetId,LayoutitWaypoint.status.saveStatus);~~ |

### 设计平台提供用户自定义方法说明

1. 在提交数据之前对vm对象的属性值进行更改，定义规则“vmid”+SubmitBefore(vm)

例如：vm1468984698064SubmitBefore(vm)。

2.在document.ready()方法后执行的函数pageDocumentReadyAfter()。

3.在document.ready()方法前执行的函数pageDocumentReadyBefore()。

## 工程管理

### 路径说明

#### 源文件路径

该值为全局选项（未提供设置），所有文件都放到D:/layouit/frame下面。该目录下存放的是设计器框架源文件，菜单中打开的文件就是该目录下文件，在保存时会先保存设计后的框架文件，然后再根据框架文件的内容生成发布文件。源文件的文件名形式为：D@$tomcat7$webapps$layoutit$test.html&pname=default，其中D@$tomcat7$webapps$layoutit表示该文件的发布目录，其中:符号由@代替，而/由$符号代替；test.html是文件名，default为工程名称。如果点击了备份功能，会把该文件备份到backup目录，文件名前面会加备份的时间点，形式为：2015\_08\_21\_11\_34\_36##D@$tomcat7$webapps$layoutit$abcd.html&pname=default

#### 发布路径

发布路径与工程进行绑定，暂只支持default工程，不可新建工程。框架会把源文件的内容按规则发布到发布路径下面。

#### 预览路径

预览路径与工程进行绑定，预览需要使用到tomcat容器，预览路径必须设置到webapps下面的某个文件夹，不然预览功能无法生效。

#### 公用JS路径

公用JS路径与工程进行绑定，用于脚本编辑器中新建、打开和保存js文件的目录，默认相对路径为“js/common/”。

### 工程管理



#### 工程信息

工程名:全局唯一，不能为空，由字母、整数和下划线组成，不可修改。

发布路径：不能为空，表单设计器生成的文件存放根路径。

预览路径：不能为空，默认与发布路径一致，预览需要使用到tomcat容器，预览路径必须设置到webapps下面的某个文件夹，不然预览功能无法生效。

公用JS路径：不能为空，相对于发布路径下，用于脚本编辑器中新建、打开和保存js文件的目录，默认相对路径为“js/common/”。

预览端口：不能为空，web容器对应的端口，默认为8080

#### 数据源

数据源列表中显示当前配置的数据源，在操作列中可编辑和删除数据源。

数据源ID:全局唯一，不能为空，由字母、整数和下划线组成，添加后该名称不能修改。

数据源名称：界面上显示数据源时的显示名称。

IP：数据源的ip。

端口：数据源端口。

URL前缀：表单设计器与ORM的接口只规划了部分路径，而前缀路径可由应用调整，所以在配置数据源时需要配置该前缀，举例说明：获取所有表名的基础接口定义为

/ metadata/tablenames，在服务器上访问路径为：<http://10.74.216.19:8080/dataservice/orm/metadata/tablenames>，这种情况下url前缀为dataservice/orm

URL 前缀可配置成带变量的动态前缀，目前表单后台会对$base\_url的前缀作相应的处理。

在设计阶段获取数据源时，如果数据源前缀中带有$base\_url时，表单后台会从config.properties 中获取base\_url 配置信息，如果配置不为空，则替换前缀中的变量为实际值，如果为空，则替换前缀变量为空。

在发布表单时，如果数据源前缀中带有$base\_url时，表单后台会从config.properties 中获取base\_url 配置信息，如果配置信息不为空，则会在default\_common.js文件中生成全局变量$base\_url，该变量值为config.properties中设置的值。在default\_common.js构建数据源信息时，会包含base\_url的变量。 配置示例如下：

|  |
| --- |
| 假设取orm的数据表的接口地址为：  <http://10.74.216.63:21180/oa/orm/metadata/tablenames>  orm设定的url前缀为：/oa//orm/，如果前缀中的”oa”可变，则在配置工程数据源URL前缀时，则可配置成如下地址：$base\_url/orm/。另外在表单类加载目录下，需要在config.properties中添加配置：base\_url = oa ，这样就完成动态前缀配置。  设计阶段获取数据源时，会将URL前缀中的变量替换为config.properties中设置的值；在表单发布时，会在default\_common.js文件中生成$base\_url的全局变量，在数据源信息变量中会包含$base\_url的变量。如下所示：  var $base\_url = "oa";  var bcp = "http://10.74.216.63:21180/"+$base\_url+"/orm/"; |

#### 设计文件

文件列表显示工程下面的所有文件，操作列可编辑和删除文件。

编辑文件可修改文件名和文件路径，该功能不会自动调整发布路径下的文件目录和文件名，如果需要发布路径下生成新的文件和目录结构需要打开设计器保存一次。

删除文件会把设计文件进行删除，发布目录下的文件不会自动清理。

新建文件时需要填写相对路径和文件名，其中如果是建到根目录则不填写相对路径，如果建立到子目录则需要填写。文件名不需要写.html，框架会自动带上文件后缀。

## 自定义数据源接入

表单设计器除默认支持orm数据源外，还支持用户自定义数据源的接入。自定义数据源接入方式通过自定义json文件来实现，json文件需要存放到表单设计器应用目录下的json-extension/datasource/目录中，在web容器启动时，会扫描此目录下的定义文件。自定义数据源结构包括如下信息：

### 数据源信息

数据源信息是一个数组对象，结构和属性如下表所示：

|  |
| --- |
| [ {  "id":"uds",//数据源id，必填  "name":"uds",//数据源名称，必填  "ip":"10.74.156.97",//数据源ip, 可选  "port":"8080",//数据源端口，可选  "app\_path":"designer", //应用名称,必填  "services"：[{...}] //服务信息,必填  } ] |

### 数据服务信息

在数据源下，包括多个服务信息，其结构如下：

|  |
| --- |
| "services":{  "id":"uds", //数据源id必填  "name":"uds",//数据源名称,必填  "service\_path":"bcp/layoutit",//服务路径，必填  "sets":[{..}] //服务数据集，必填  "definitions":{...} //数据集中引用对象，如果数据集中包括引用对象，则必须要添加此对象定义。  } |

### 数据集信息

1、在数据服务下包括多个数据集，其结构定义如下：

|  |
| --- |
| "sets": [{  "name":"demo\_user",//数据集名称，必填  "set\_path":"",//数据集地址  "fields":[{...}],//数据集字段  "post\_in":[{...}],//数据集添加字段定义  "put\_in":[{..}],//数据集修改字段定义  "get\_out":[{...}],//数据集查询字段定义  "operations":[]//数据集操作定义  }] |

数据集下字段定义用于组件件字段绑定，在数据集结构定义中，fields,post\_in,put\_in和get\_out中必须填写其中一项，此数据集定义才生效。

2、数据字段定义支持嵌套对象，支持嵌套对象结构为：1:1:1..1,1:1...1:n，字段结构定义如下：

|  |
| --- |
| "fields": [  {  "name":"userid",//字段名称  "type":"int",//字段类型  "length":"11"//字段长度  },  {  "name":"demo\_user\_blog",//字段名  "type":"object",//字段类型，object是引用对象，需要在引用对象中定义此对象.  "$ref":"blog"//引用对象名  },  {  "name":"demo\_user\_roles",//字段名称  "type":"array",//字段类型，为数组时，需要到引用对象中定义  "$ref":"roles"//引用对象名称  }  ] |

post\_in,put\_in和get\_out结构和fields结构一致。

3、operations 是数据集对应的操作定义

针对数据集的操作，系统默认包括增、删、改查四种操作，根据开发需要，用户可添加新的操作方法或者修改默认操作方法的参数，操作定义结构如下：

|  |
| --- |
| "operations":[  {  "method":"POST",//数据添加  "url":"",//服务url  "getUrl":"",//用户自定义获取服务url函数  "submit":"",//用户自定义提交函数  "success":"",//成功回调  "error":""//失败回调  },  {  "method":"PUT",//数据修改方法  "url":"",  "getUrl":"",  "submit":"",  "success":"",  "error":""  },  {  "method":"DELETE",//数据删除方法  "url":"",  "getUrl":"",  "submit":"",  "success":"",  "error":""  },  {  "method":"GET",//数据查询方法  "url":"",  "getUrl":"",  "submit":"",  "success":"",  "error":""  }  ] |

### 引用对象信息

在数据集中字段定义如果包含object或array类型的字段，则需要在引用对象中定义,如果引用对象中也包括object或array，同样需要将该对象定义成definitions的子对象， 结构定义如下：

|  |
| --- |
| "definitions":{  "blog":{//子对象名称  "properties":[//字段属性  {  "name": "id",//字段名称  "type": "int",//字段类型  "length": "11"//字段长度  },  {  "name":"demo\_user\_article",//字段名称  "type":"object",//字段类型  "$ref":"article"//引用对象  }  ]} ,  "article":{  "properties":[  {  "name": "id",  "type": "int",  "length": "11"  },  {  "name": "userid",  "type": "int",  "length": "11"  }]  }  }] |

### 自定义json数据源示例



## 数据集设置

### 简介

数据集设置是为满足用户能根据自身业务需求通过页面操作灵活创建数据集，并且能对数据集进行修改，删除的功能。

### 数据集列表

点击表单设计器页面工具栏按钮中的“数据集设置”弹出一个新的tab页，新的tab页上显示的就是数据集的列表，数据集列表分为上下两部分，上部分为新增数据集操作栏，输入不重复的数据集名称（点击新增时会验证数据集名称是否重复），选择数据源及需要作为主表的主表名，然后点击“新增数据集”按钮后弹出新的tab页进行操作；下半部分就是数据集的列表，列表当中会显示所有数据源中定义的数据集，在列表的最后一列“操作”列，可点击“修改”、“删除”图表，对对应的数据集进行操作。

### 新增、修改页面

新增、修改页面就是对数据集具体操作的页面，在此页面中，就是对数据集具体的定义，首先必须对主从表的关系进行设置，点击页面上的“关系设置”按钮弹出对话框对主从表关系进行设置，在弹出窗口的页面上有四个下拉框，分别是关系，从表，主表字段，从表字段。关系目前有两个选项，分别为1：1、1：N，即主从表之间的对应关系，从表里的选项是除主表以外的所有当前数据库当中的表，当选择了从表之后，从表字段会根据选择的从表生成对应的字段选项，主表字段的选项就是之前选择的主表的字段，当四个选项都选择好过后，点击右下角的添加，则会将对应的选项加入到下方的关联字段列表当中。在这里列表的显示有以下的规则：假设列表当中已经有记录，如果是同一张从表，选择的主表字段没变，但是将从表字段换为其他字段了，此时点击添加按钮，将对对应的主表字段所在的行记录的从表字段进行更改，同样，还是在一张表当中，主表字段改变，从表字段不变，那么就将改变对应的从表所在行的主表的记录；同样选择的同一个从表当中，选择的主表字段假如说已存在于列表的第一条数据，从表字段存在于列表的第二条数据当中，此时，如果点击添加的话，那么会将第一、二条两条记录都删除并且插入一条以当前选择的主表字段及从表字段组成的数据。另外一个场景，如果列表当中已经有数据，而此时选择的从表与列表中的从表不同，但是选择的主表字段却已经存在于列表当中不同从表的记录当中，此时点击添加，将无法添加成功，页面会提示“主表字段已关联”。当所有的关系都已经设置完毕过后，点击弹窗的确定按钮，将会根据所选择的从表在“数据集字段选择”页面生成相应的从表字段选择页面，根据业务需求选择需要查询显示的主从表字段，操作很简单，只需在前面的选择框点击选中即可，而如果此时如果什么都不选择而点击页面下方的确认按钮，页面将会提示“请选择主表字段”，选择了主表字段过后而没有选择从表字段或者从表当中有一张表没有选择字段，那么在点击确定时都会提示“请选择从表字段”。在一切都选择完成过后，点击确认则数据集保存成功，并且退回至列表页面。过后，便可以通过“工具栏按钮”的“新建”当中的“按模版创建”中选择使用新增或修改好的数据集了。

## 工具栏按钮

### 新建

新建文件，下方包含“确定并打开”和“确定”。其中“确定并打开”框架生成文件后并打开文件进行设计。而“确定”后不会打开设计界面。

#### 按模版创建

点击“新建”过后，再点击“按模版创建按钮”，弹出窗口，进入到关系模型选择页面，选择所要创建的关系模型图，点击确定后进入数据源与多表关系的选择页面，确定选择了数据源与多表关系后，会在当前窗口页面的下方展示出主从表的表名、对应关系、以及各自所要展示的字段，再次点击确定后，在表单设计器中心设计面板，将会按照对应的主从表关系生成以表单或表格组成的页面，而字段则会默认以不同的形式做展示；比如1：1关系生成的就是表单页面，当中的字段默认会以文本框做展示，并且标签也会在文本框前方以字段名做展示。而以1：N关系生成的就是一张表格，字段则绑定在表格中的列做展示。

### 打开

打开对话框显示设计文件列表，可双击表格行或者选中行后点击“确定”按钮打开文件进行设计。

### 保存

保存当前源文件并把设计好的文件发布到设置的发布目录中。

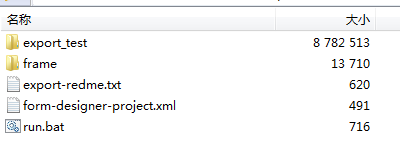
### 导出

设计文件：表单设计器保存的框架文件，后面在有新需求时可通过表单设计器打开该文件修改界面。建议把该文件上传到svn，供后续需求开发使用。

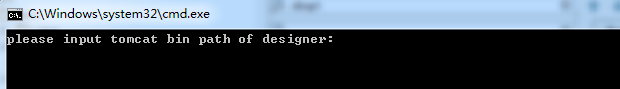
页面文件war：表单设计器生成出来可以使用的页面文件，该目录放到tomcat webapps中可直接部署。

后台服务：如果部署了orm，可以把orm一起导出，导出后“页面文件war”不仅可以显示界面还可以操作数据库。

导出文件如下



其中运行run.bat可把导出的内容重新导入到tomcat中，run文件运行效果：



必须输入tomcat/bin所在目录。

### 预览

预览时会重新保存一遍文件，在预览前不需要手动保存一遍。如果在工程管理中设置的预览路径和发布路径不一致，预览时会把发布路径下生成的当前页面文件复制一份到预览路径下。

### 工程管理

详情参见 工程管理 模块。

### 备份

备份当前设计的框架文件和发布文件。

框架文件放到D:/layoutit/frame/backup中，文件名中“##”号前面部分表示备份时间。

发布文件是放到发布路径下的/backup中，文件名中“##”号前面部分表示备份时间。

### 脚本编辑器

请参见 脚本编辑器 模块。

### 数据集设置

详情参见 数据集设置 模块。

### 使用手册

表单设计器最新使用手册下载。

## 组件使用

### 全局属性

#### Id、Name

全局属性，属性值可修改，属性对应具体组件的id和name两个html属性。

**id和name字段调整后才能设置其他属性**，其余属性都是依赖id和name的，在id/name变化未更新这些属性 。

#### 可编辑性

在表单新增、修改、查看时该组件是否可编辑，选中表示可编辑，不选中表示灰化。默认新增、修改可编辑，查看灰化。

只有表单支持的组件才有该选型，如果组件是用于表格查询时该属性不生效。

#### 绑定字段

组件绑定的数据库字段，如果涉及的页面不使用ORM提供的功能，可以不绑定字段。只有表单支持的组件才有该选型，如果组件是用于表格查询时该属性不生效。

### 布局

布局比例： 布局组件显示比例。布局器设计为12格，当布局器中未添加其他组件时可修改该属性值，例如设置为4,4,4表示平均分配的3列，6,6表示平均分配为2列。

数据源：布局组件操作的数据表或者数据集。只有顶层布局组件才能设置数据源，顶层布局组件里的布局组件都继承该数据源。

布局器里的组件未绑定字段时可以设置和修改该选项。在选择数据源后，下方表格会显示该数据源字段和数据类型。

### 标签

显示文本：标签显示的文本内容。

对齐方式：默认为右对齐，如果有其他需要时需要设置该选项。

### 文本框、文本域

默认值 ：文本框、文本域默认显示内容。

提示信息 ：在没有输入值时，placeholder显示内容，文本框独有。

高度 ：文本域高度，以Rows为单位，文本域独有。

组件类型 ：分为文本框和密码框，文本框独有，默认为文本框。

是否为空 ：在新增和修改表单时，内容是否允许为空。

长度范围 ：输入字符的长度范围。

验证类型：分为身份证号码、邮箱、手机号码、ipv4地址、整数、浮点数和自定义验证方式。

自定义验证脚本

|  |
| --- |
| notEmpty: {message: 'The first name is required and cannot be empty'}//不为空+消息自定义。  stringLength: {min: 6, max: 30, message: 'The username must be more than 6 and less than 30 characters long'} //长度验证+消息自定义  regexp: {egexp: /^[a-zA-Z0-9\_\.]+$/,message: 'The username can only consist of alphabetical, number, dot and underscore' }//正则表单式验证  different: {field: 'password,confirmPassword', message: 'The username and password cannot be the same as each other'} //输入框内容相同验证  emailAddress: { message: 'The input is not a valid email address'} //邮箱验证  date: {format: 'YYYY/MM/DD', message: 'The birthday is not valid'}//时间验证 |

关于验证详细资料参见：<https://github.com/psyked/bootstrapvalidator>

### 下拉框

模式：单选下拉框和多选下拉框两种模式。

默认选中项：下拉框默认选择项。当下拉框模式是多选下拉框时，多个默认值之间用逗号分隔。

选项 ：下拉框的选项内容，形式为name:value，多个选项之间换行。

动态选项URL：下拉框URL，在页面加载时从服务端获取数据，服务端返回数据后加载到下拉框中。服务端返回数据格式：

//直接返回数组数据：

[{"key":"name1","value":"value1"},{"key":"name2","value":"value2"},{"key":"name3","value":"value3"},{"key":"name4","value":"value4"}]

数组数据封装到data对象中：

{"data":[{"key":"阿坝","value":"阿坝"},{"key":"巴中","value":"巴中"}]}

参考代码：http://10.75.8.170/svn/ZXUEP\_BCP\_REPOS/trunk/src/form-designer/src/main/java/com/zte/iui/layoutit/service/LayoutitRestTest.java中getSelectDynamicData方法

动态下拉框加载url的方法在js/page/ layoutitPageLoad.js中，方法名为：initSelectByDynamicURL，如有问题可先自行分析。

脚本对象：界面初始化时把对象内容加载到下拉框中，**动态选项URL与脚本对象只能填入一个**。对象内容形式为name:value的数组。

表单设计器框架提供通过ORM查询表格数据填充到动态下拉框中。

|  |
| --- |
| 脚本组件中编写代码  var zrdepartOptions=[];//定义全局变量  getSelOptionForm("zrdepart","departname","id", zrdepartOptions);  /\*\*  \* 根据表名，显示名、值等信息加载下拉框数据。<br />  \* eg: var zrdepartOptions=[];  \* getSelOptionForm("zrdepart","departname","id", zrdepartOptions);  \* @param table 需要查询表名  \* @param name 显示名称列  \* @param value 显示值列  \* @param options 需要加载的数组对象，该对象必须外部定义。  \* @param condition 过滤条件，如果没有条件可以不传入该对象或者传为"undefined"  \*/  getSelOptionForm("表名","显示列名","值列名", 全局数组对象); |

动态加载脚本对象的方法在js/page/ layoutitPageLoad.js中，方法名为：initSelectByDataScriptObject，如有问题可先自行分析。

注意：

1）脚本对象加载方式，脚本对象采用同步顺序加载方式，目前不支持异步加载。

2）脚本对象加载顺序，对象加载应该在initSelectByDataScriptObject方法前加载，才能被表单系统识别。

4）脚本对象默认加载的方法：如果加载时表单系统发现数据对象为空，则会默认调用用户定义的方法：loadCustomDataScriptObject();

5）脚本对象作用域为全局对象。

### 单选框

默认选中项： 单选框默认选中项

选项：单选框的选项内容，形式为name:value，多个选项之间换行。

脚本对象：界面初始化时把对象内容加载到单选框中，选项与脚本对象只能填入一个。对象内容形式为name:value的数组，可参考下拉框脚本加载方式。

动态加载脚本对象的方法在js/page/ layoutitPageLoad.js中，方法名为：initRadioButton，如有问题可先自行分析。

### 日期

默认值 ：日期组件默认显示内容。

提示信息 ：在没有输入值时，placeholder显示内容。

是否为空 ：在新增和修改表单时，内容是否允许为空。

验证类型：选项为日期、日期时间和自定义。当为自定义时可以定义当前组件与其他组件的时间对比，自定义时间对比说明：

1）日期验证方法dateLessThan,当前组件小于field组件,如：dateLessThan:{field:'name'[,...]}

2) 日期验证方法dateGreaterThan,当前组件大于field组件,如：dateGreaterThan:{field:'name'[,...]}

注意：需要验证时日期组件都需要增加自定义验证，filed可以为多个，每个name之间用逗号分割。

### 上传

URL ：上传组件服务端响应url

文件格式 ：允许上传文件的后缀名，多个之间以英文逗号分割。

是否非空 ：在新增和修改表单时，内容是否允许为空。

上传组件需要服务端开发代码实现，文件上传到服务端后需要返回文件上传的路径，该内容组织到data的json对象中。

### 按钮

显示名 ：按钮的显示名称。

按钮类型：选项为提交、重置和自定义。

表单提交时使用提交。

提交前参数过滤回调方法：回调函数必须以vmid+” ParameterFilterCallBack”作为函数名，参数为data，保护所有参数。新增时只有columns参数，组织形式为{columns: newmodel}，修改时会增加condition参数，组织形式为{columns: newmodel, condition: condition}。

Ajax请求success后回调方法：提交成功后调用回调函数处理额外信息，回调函数必须以vmid+” SuccessCallBack”作为函数名，服务端返回的结果作为参数，参数只有一个。vmid是表单提交的当前vmid，通过查看生成的js文件进行查找，搜索avalon.define({$id:字符串，后面的字符串即可vmid的值。该方法定义到脚本组件中。

Ajax请求error后回调方法：提交成功后调用回调函数处理额外信息，回调函数必须以vmid+” ErrorCallBack”作为函数名，无参数。

表单提交是调用avalon定义的方法submit，该方法的ajax请求为同步请求，返回值包含ajax的请求状态，返回结果形式为：

{status:1, message:""} --成功

{status:0, message:"validate fail. "} --失败，message描述失败的详细信息

表单需要提供重置功能时，选择类型为重置，非表单组件使用时，不能选择按钮类型为重置。

其他情况使用自定义，自定义时需要在事件中编写按钮click事件方法。

### 远程表格

远程表格提供服务端分页，表格数据是使用的ORM接口。

表格高度:控制表格高度，默认500px,输入数字

每页显示条数 ：表格每页显示条数

表格模式：分为普通表格、单选和多选表格，其中单选表格会在每行前面加单选按钮，多选表格会加多选按钮。

表格头对齐方式：表格头对齐方式，包含靠左、居中、靠右三个选项。

表格列对齐方式：表格列对齐方式，包含靠左、居中、靠右三个选项。

自定义工具栏:表格自定义工具栏按钮，主要属性有按钮名称、按钮样式、按钮事件，如下：

[{"title": "按钮1","icon":" glyphicon glyphicon-cog","clickfunction": "changeBtn('aa','bb')"},

{"title": "按钮2","icon":" glyphicon glyphicon-cog","clickfunction": " changeBtn('aa','bb')"}]

高级属性中的列定制

标题：表格头显示文本。

字段：表格内容显示字段。

格式化：表格列格式化方法，这里只写方法名，方法的具体实现放到脚本组件中编写。

框架提供了三个方法

排序：该列是否排序，如果不排序则不勾选。

隐藏：列是否隐藏，勾选表示需要隐藏。

主键：该字段是否作为主键，在修改和删除时需要使用主键字段作为条件。

列顺序：可以点按钮切换列之间顺序。

默认条件：在默认显示数据时的过滤条件，例如表格只显示a字段>5的值，需要选择该字段并设置条件。 --已废弃。

查询条件：勾选后，在表格工具栏中会加入条件过滤，过滤方式和选择的保持一致。

注意：

表格组件的所有修改都必须点击应用按钮才生效。

框架提供的表格接口方法参见 [4.3.5](#_表格接口) 。



在勾选编辑时表示该列可编辑，格式化中选择编辑需要的组件，只有选择文本框时才需要选择验证类型，验证类型提供了通用验证类型和自定义两种，选择自定义时需要定义js变量存储正则表达式。js变量命名和内容定义规则：

|  |
| --- |
| /\*\*  \* 提供用户自定义验证，自定义变量格式如下，message为验证不通过时的提示信息  \* var table$tablieid$field = {message:"" ,regex:/^\d+$/}  \*/  /\*\*  \* 提供用户自定义变量，实现表格下拉框数据动态添加\*  var table$tablieid$field = {  data:  [  {text:" ",value:" "},  {text:" ",value:" "}  ]  }  \*/ |
|  |

不勾选编辑时，格式化中填写格式化函数，格式化函数示例

|  |
| --- |
| /\*\*  \* 表格列格式化函数，显示效果为输入框  \* @param value  \* @param row  \* @param index  \*/  function formatTableText(value, row, index) {  this.isupdate = false;  var val = value;  if(value == undefined) {  val = "";  }  if(this.isupdate){  return '<input type="text" value="' + val + '" class="form-control" onblur="editorTableUpdate(' + index + ', this, \'' + encodeURIComponent(JSON.stringify(row)) + '\', \'' + this.field + '\', \'' + this.tableId + '\')" />';  }  return val;  } |

### 本地表格

本地表格的数据是通过js加入的，和服务端没有任何关系，在web客户端处理数据时使用。本地表格默认增加操作列，操作列提供删除本行数据功能。

表格高度:控制表格高度，默认500px,输入数字

表格模式：分为普通表格、单选和多选表格，其中单选表格会在每行前面加单选按钮，多选表格会加多选按钮。

高级属性：

添加按钮：添加表格显示列。

删除按钮：删除已勾选的表格列。

注意：

表格组件的所有修改都必须点击应用按钮才生效。

框架提供的表格接口方法参见[4.3.3](#_表格接口)。

### 对话框

对话框只能拖拽到隐藏容器中。

宽度 ：对话框宽度。

高度 ：对话框高度。

静态对话框显示内容(支持HTML)：静态对话框显示内容，如果是打开的远程url对话框，则不需要填写该部分内容。

对话框footer：对话框是否添加footer。

如果对话框加载的url已经提供了按钮操作则不需要添加footer。

如果对话框加载的url没有添加按钮，则需要添加页脚。添加页脚时默认增加关闭按钮，只有勾选“添加确认按钮”时才会添加确认按钮，确认按钮click内容添加到下面的输入框中。

对话框方法调用参见：[4.3.4](#_对话框接口)

### 脚本

脚本文件相对路径 ：该页面引用的js文件，路径为相对路径。

### 树

#### 树组件属性配置

节点名称： 树节点显示名称对应的数据字段名。

节点ID： 树节点ID对应的数据字段名。

父节点ID： 树节点父节点ID对应的数据字段名。

节点提示： 树节点提示对应的数据字段名。

树类型： 普通类型（树节点不能选择）、节点单选和多选三种类型。

URI： 树的数据来源。

1）使用ORM提供接口返回表格数据时为了不区分表格返回数据结果一样，树数据返回结果与表格返回结果一致，返回的属性名必须和界面上设置的属性名保持一致。

{"total":5,"rows":[

{"NodeID":0,"NodeURL":null,"ParentNodeID":null,"Remark":"1","NodeTitle":"中兴通讯","NodeName":"中兴通讯"},

{"NodeID":1,"NodeURL":null,"ParentNodeID":0,"Remark":"2","NodeTitle":"中兴成都分公司","NodeName":"中兴成都分公司"},

{"NodeID":2,"NodeURL":null,"ParentNodeID":0,"Remark":"3","NodeTitle":"中兴西安分公司","NodeName":"中兴西安分公司"},

{"NodeID":3,"NodeURL":null,"ParentNodeID":null,"Remark":"4","NodeTitle":"中国电信","NodeName":"中国电信"},

{"NodeID":4,"NodeURL":null,"ParentNodeID":null,"Remark":"5","NodeTitle":"中国联通","NodeName":"中国联通"}]}

2) 自定义开发代码返回树结果时

结果放到data对象中，data可以为json字符串格式的数据，也可以是js数组对象（建议使用js数组），格式如下：

[{id: "101",name: "aaa",parentId: "12"},{id: "102",name: "bbb",parentId: "15"}...]

#### 树组件事件

节点绑定前事件

**描述**：树节点绑定前执行该事件。

**事件参数**：node （树节点对象）

**示例：**设置节点图标

|  |
| --- |
| **function** *ztreeNodeFormatForOrg*(node){  **if**(node.dimensionType === 1){  node.**icon** = **"img/ztree/org.gif"**;  }**else**{  node.**icon** = **"img/ztree/project.gif"**;  } } |

### 分隔栏

标题 ： 分隔栏显示文本。

分隔线颜色： 分隔栏颜色。

分隔线图标：分隔栏显示图标。

### iframe

高度(支持百分比) ：iframe高度

宽度(支持百分比) ：iframe宽度

src：iframe的src设置，路径为当前设计页面的相对路径，不能写/开头的路径。

### Tab

选项:Tab组可以包含多个Tab,格式为:

[{"title": "Tab名称","targetid": "Tab内容ID"},

{"title": "tab2","targetid": "vm1437382282986"}]

### Collapse可折叠

选项: Collapse组可以包含多个collapse,格式为:

[{"title": "collapse名称","targetid": "collapse内容ID"},

{"title": "Col#3","targetid": "vm1437440027908"}]

### 按钮组

选项: 按钮组可以包含多个button,格式为:

[{"title": "按钮1","icon":" glyphicon glyphicon-cog","clickfunction": "changeBtn('aa','bb')"},

{"title": "按钮2","icon":" glyphicon glyphicon-cog","clickfunction": " changeBtn('aa','bb')"}]

### 图形组件

#### 柱状图

标题：柱状图的标题

图例：图中柱体或连线所代表的意义

提示框：鼠标悬浮时的信息提示。

工具箱：对柱形图进行操作的各个工具集合。

高级属性

数据接口：通过URL访问数据接口获取数据

自定义URL：判断是否是来自外部的数据接口

图例名称：对通过URL获取的数据在图中显示进行定义。

X轴：当获取到数据时，会在X轴下拉选项中获取到对应的展示项，如果选择其中一项，则Y轴默认选中另一项。

数据系列：输入URL以及图例名称后生成，可对在图例中定义的数据进行样式的选择。

#### 折线图

各个属性均与柱状图相同。

#### 饼图

高级属性中的标题、提示框、工具箱、图例与柱状图相同，不同的是饼图中以上是以标签收起，可点击标签展开后进行设置。

Series：驱动图表生成的数据内容数组

数据接口：通过URL访问数据接口获取数据

名称：当获取到具体数据后，会在数据接口输入栏下方，生成一个带数据名称标题的面板，当中的名称即数据定义名称

字段：在饼图中显示具体的字段。多选可删除

圆心坐标：圆心所在的位置。支持绝对值（px）和百分比

半径：圆心的半径。支持绝对值（px）和百分比

#### 多图联动

基本属性与柱状图类似

高级属性

图表布局：可对图表个数在一行当中显示做调整。

图表数量：图表显示个数。

数据接口：通过URL访问REST接口获取数据

图例名称：对通过URL获取的数据在图中显示进行定义

X轴：当获取到数据时，会在X轴下拉选项中获取到对应的数据展示项，如果选择其中一项，则Y轴默认选中另一项。

各图表属性设置：当以上的设置完毕后，会根据所输入的图表个数生成对应图表的属性设置。

### ~~路标导航~~

~~用于当页面内容较多出现滚动条时，快速导航到绑定路标属性的元素上去。~~

~~此组件支持两种状态标识，0:初始状态；1:保存状态。开发人员可调相关API改变组件节点状态。~~

**~~导航配置~~**~~：配置需要绑定路标元素的属性,包括如下信息：~~

~~targeteId:路标组件关联的容器Id，必填。~~

~~name:路标导航上显示的名称，必填。~~

~~Icon:路标组件显示的节点图片路径，可填。~~

~~initStatusIcon：路标节点初始状态时图标的路径，不填时节点以红色作为背景色。~~

~~saveStatusIcon：路标节点保存状态时图标的路径，不填时节点以红色作为背景色。~~

~~配置示例如下：~~

|  |
| --- |
| ~~[{"targetId":"layout1447759614814","name":"step1","icon":"","status":0,"initStatusIcon":"css/easyui/icons/clear.png","saveStatusIcon":"css/icons/clear.png"},~~  ~~{"targetId":"layout1447759617120","name":"step2","icon":"","status":0,"initStatusIcon":"css/easyui/icons/clear.png","saveStatusIcon":"css/icons/clear.png"}]~~ |

### 移动端-购物车

为移动端开发购物车功能提供的组件，该组件需要配置对象名和结算事件两个属性，购物车是根据对象中的内容生成购物车页面的，页面通过全局方法initShoppingCartData初始化该数据，生成的购物车页面会自动调用该方法。

下面实例中，在对象名中输入的值为products,方法initShoppingCartData初始化该对象。

|  |
| --- |
| function initShoppingCartData() {  products = [  {  //店铺信息  "shop": {  "id": "327741", //店铺编号，全局唯一  "name": "成都汽车配件专供", //店铺名称  "link": "http://10.74.156.97:8080/udsmall/goods\_98485.htm" //店铺链接，可为相对路径和绝对路径  },  //店铺下加入购物车的商品列表  "products": [  {  "id": "984951", //商品编号，全局唯一  "name": "东风汽车配件 天龙新款驾驶室", //商品名称  "link": "http://10.74.156.97:8080/udsmall/goods\_98495.htm", //商品链接，可为相对路径和绝对路径  "price": 32000, //商品销售价格  "number": 1, //加入购车中的数量  "stock":10, //商品总库存  "img":"http://10.74.156.97:8080/udsmall/upload/store/32774/2016/05/11/7be42903-9cf1-469a-9b90-65b3f1e679bb.png\_small.png", //商品缩略图  "color":"白色", //选择的商品颜色，可不填写该属性  "size":"大号" //选择的尺码，可不填写该属性  },  {  "id": "984912",  "name": "东风汽车配件 东风天龙大力神挂车制动阀3522Z07",  "link": "link",  "price": 175,  "number": 2,  "stock":20,  "img":"http://10.74.156.97:8080/udsmall/upload/store/32774/2016/05/11/48fefc04-cd62-474f-b0b9-8ba8b4e7c2ed.jpg\_small.jpg",  "color":"白色",  "size":"大号"  }  ]  },  {  "shop": {  "id": "327742",  "name": "成都汽车配件专供",  "link": "http://10.74.156.97:8080/udsmall/goods\_98485.htm"  },  "products": [  {  "id": "984953",  "name": "东风汽车配件 天龙新款驾驶室",  "link": "http://10.74.156.97:8080/udsmall/goods\_98495.htm",  "price": 32000,  "number": 1,  "stock":20,  "img":"http://10.74.156.97:8080/udsmall/upload/store/32774/2016/05/11/4650030f-d7bb-4ab4-a5b4-6b43e04552ac.jpg\_small.jpg",  "color":"白色",  "size":"大号"  },  {  "id": "984914",  "name": "东风汽车配件 东风天龙大力神挂车制动阀3522Z07",  "link": "link",  "price": 175,  "number": 2,  "stock":20,  "img":"http://10.74.156.97:8080/udsmall/upload/store/32774/2016/05/09/13639523-8fb8-4af6-b4ec-b38bc3c6f6ad.jpg\_small.jpg",  "color":"白色",  "size":"大号"  }  ]  }  ];  } |

结算事件中可填写函数，也可填写实现内容，建议填写函数，函数参数为data，该参数的形式与上述对象一致，只是把内容调整为用户选择和修改后的内容。

## orm接口

所有接口前面都会加上数据源配置ip、端口和uri前缀。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | Path | 提交方式 | 参数 | 返回结果 |
| 获取所有表 | / metadata/  tablenames | GET | 无 | {"tablenames":"a\_layout a\_layout\_person a\_layout\_tree a\_layout\_trouble act\_evt\_log"} |
| 查询表字段信息 | /metadata/table/a\_layout\_person | GET | 表名参数通过url传递，url中的a\_layout\_person是表名。 | [{"character\_maximum\_length":50,"column\_name":"StaffID","data\_type":"string"},{"character\_maximum\_length":50,"column\_name":"Name","data\_type":"string"}] |
| 新增多行 | multi/table/a\_layout\_trouble |  | 表名参数通过url传递，body参数{"columns":[{"name":"名称","alarmlevel":"major","siteid":2},{…}]} | 成功返回：success  失败返回：fail |
| 新增 | /table/a\_layout\_trouble | POST | 表名参数通过url传递，body参数{"columns":{"name":"名称","alarmlevel":"major","siteid":2}} | 成功返回：success  失败返回：fail |
| 修改多行 | multi/table/a\_layout\_trouble | PUT | 表名参数通过url传递，body参数records:[{"columns":{"id":4,"name":"名称1","alarmlevel":"minor",},"condition":{"cname" : "id","value" : 1,"compare" : "="}},…] | 成功返回：success  失败返回：fail |
| 修改 | /table/a\_layout\_trouble | PUT | 表名参数通过url传递，body参数{"columns":{"id":4,"name":"名称1","alarmlevel":"minor",},"condition":{"cname" : "id","value" : 1,"compare" : "="}} | 成功返回：success  失败返回：fail |
| 删除 | /table/a\_layout\_trouble | DELETE | 表名参数通过url传递，body参数{"condition":"or":[ {"cname" : "id","value" : 1,"compare" : "="},{"cname" : "id","value" : 2,"compare" : "="} ]} | 成功返回：success  失败返回：fail |
| 查询 | /table/a\_layout\_trouble | GET | 表名参数通过url传递，body参数{"columns":[{"cname":"id"},{"cname":"name"},{"cname":"description"}],”isDistinct”:”true”,"condition":"or":[ {"cname" : "id","value" : 1,"compare" : "="},{"cname" : "id","value" : 2,"compare" : "="} ]} | {"total":2,"rows":[{"id":"1","name":"name1","desc":"desc1"},{"id":"2","name":"name2","desc":"desc2"}]}  数据为空返回形式：{"total":0,"rows":[]}  注意：在不传isDistinct时，服务端处理为isDistinct=false。 |

## 组件实例

### 数据嵌套对象demo使用

1. 首先在数据库建立四张表分别为： bfd\_user、bfd\_user\_company、bfd\_user\_position、 bfd\_user\_roles，关系为bfd\_user嵌套bfd\_user\_company、bfd\_user\_roles，bfd\_user\_company嵌套bfd\_user\_position（它们中间以userid这个字段链接,此四张表的建表sql路径：form-designer\src\main\webapp\example\db\database.sql）
2. 此时需将form-designer\src\main\webapp\example\datasource\datasource.json复制到designer\json-extension\datasource路径下的datasource.json使文件内有相应嵌套关系的结构。
3. form-designer\src\main\webapp\example\designer-file\nesting\_object\_demo.html&pname=default&version=1.0复制设计文件到designer-frame-files，目录下此时打开设计界面，页面初始化会加载嵌套数据文件datasource.json和设计文件nesting\_object\_demo，打开设计文件会出现相应的demo和对应的嵌套数据源。
4. 此demo的简单绑定数据说明，依次按层级绑定，最外层绑定bfd\_user表，第二层绑定bfd\_user\_company和bfd\_user\_roles(此时的是以本地表格组件绑定的，为array对象)，第三层绑定bfd\_user\_position。
5. 然后对相应的字段进行组件绑定，最后设置提交按钮。

## 表单设计器ajax通用接口

通用的ajax接口文件为designer-orm-base.js文件在designer\layoutit\js\common 目录下。

先将此js文件引入

这个接口中主要提供的方法有：增(insert)删(delete)改(update)查(query)以及通用 的ajax请求接口。直接使用$.designerOrmBase.方法即可调用。例如：

($.designerOrmBase.insert()).

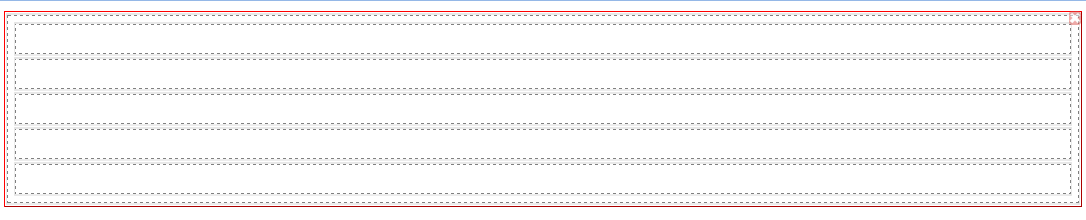
# 使用规范

## 布局组件

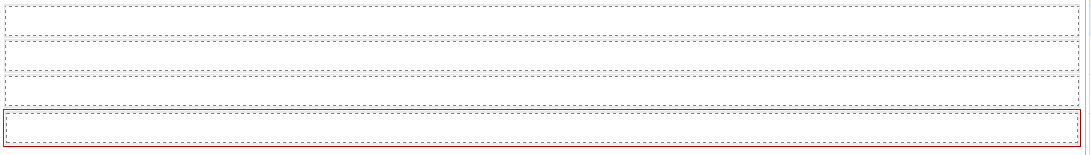
### 布局组件尽量嵌套使用，不要平铺使用。

外层一个布局组件，布局比例为12，内层多个布局组件，布局比例按需求调整；如果存在多个数据源选择情况才需要加入外层布局器。

正确例子：外层布局组件嵌套多个里层布局器。



错误例子：都是外层布局组件，没有做嵌套。



### 组件隐藏通过布局组件来实现，不能直接在样式中编写display:none

编写display:none属性会把布局器上的组件隐藏，无法再进行编辑。

# 问答

### 表单设计中如何控制组件的灰化和使用？

通过url中的参数来控制表单是新增/修改和查看，参数分别为operator=add/ operator=edit/operator=view，例如输入框A定义为新增可编辑、修改和查看不可编辑时，url中operator=add时是可以编辑的，而operator=edit/operator=view这两种情况是灰化的。

### 为什么在表单查看时所有按钮都不见了？

框架认为在表单查看时是不应该有任何操作的，在operator=view时会隐藏界面上所有按钮。