**DeWeb开发指南**

（2024-04-02）

# 简要介绍

DeWeb是一个可以直接将Delphi程序快速转换为网页应用的工具！

使用DeWeb, 开发者不需要学习HTML/CSS、JavaScript、Java、PHP、ASP、C#等新知识，用Delphi搞定一切！

支持开发web、公众号和小程序！

DeWeb开发的网页支持所有客户端，包括手机、平板等。

网站: <http://www.delphibbs.com>

# 支持控件

* 支持所有数据连接组件(ADOConnection/FDConnection/UniConnection)等；
* 支持所有数据表组件（Query/Table）;
* 支持所有三层中间件；
* 支持Button
* 支持Calendar
* 支持Chart，采用Echarts库，功能更加丰富
* 支持CheckBox
* 支持ComboBox
* 支持DateTimePicker
* 支持Edit
* 支持FlowPanel
* 支持GridPanel
* 支持GroupBox
* 支持HeaderControl，多种样式
* 支持Image
* 支持Label，另外可以用作验证码
* 支持ListBox
* 支持ListView
* 支持MainMenu
* 支持MediaPlayer
* 支持Memo，另外有多种用途
* 支持Panel，另外有多种用途
* 支持PageControl
* 支持RadioButton
* 支持SpinEdit
* 支持Shape(用于扫描二维码)
* 支持SpeedButton
* 支持StringGrid
* 支持Timer
* 支持TreeView

# 入门教程

## 开发版本支持

1. 目前支持Delphi 10.4.2 / 11.3 /12
2. DeWeb默认使用Delphi 12 编译，推荐使用此版本开发！

## 选用服务程序版本

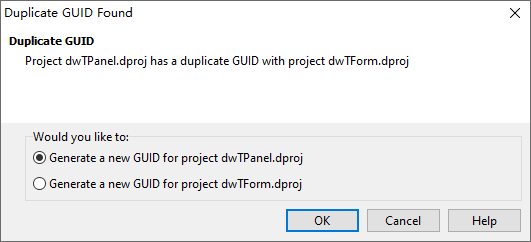
DeWeb的服务程序为DeWebServer.exe（简称**DWS**）

1. 如果Delphi版本是12，请忽略这一步；
2. 如果Delphi版本是10.4.2，请先删除Source\Runtime\DeWebServer.exe。然后重命名Source\Runtime\DeWebServer1042.exe为Source\Runtime\DeWebServer.exe；
3. 如果Delphi版本是11.3，请先删除Source\Runtime\DeWebServer.exe。然后重命名Source\Runtime\DeWebServer113.exe为Source\Runtime\DeWebServer.exe；

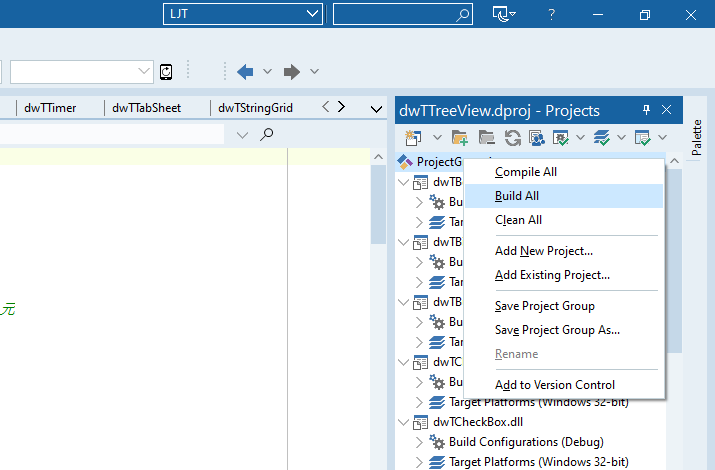
## 编译支持控件

1. 打开Source\dwVcls\中所有dpr

如果遇到以下情况，选择“OK”按钮

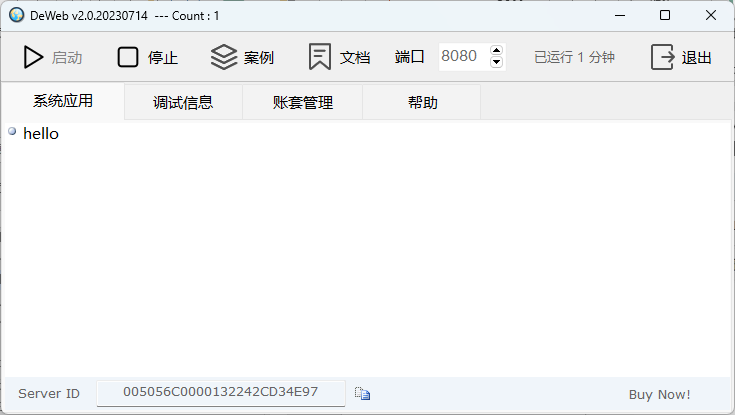


1. Build all（构建所有），注： 不是Complile all!!!

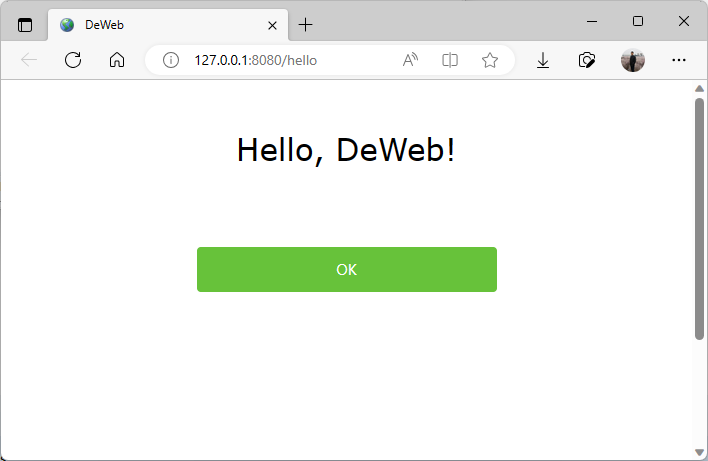


## 第一个案例：Hello,World!

1. 打开Source\Demos\01 hello目录中的hello.dpr
2. Run(F9)编译运行，即可以直接打开服务器程序DeWebServer.exe



1. 双击hello,即可打开当前应用



如果浏览器未显示上述页面，建议改用现代浏览器（Edge/Chrome/Firefox，推荐Edge），国产浏览器请使用极速模式。

# 数据库开发

## 4.1 基础案例

1. 打开Source\Demos\02 dbhello目录中的dbhello.dpr
2. Run(F9)编译运行，即可以直接打开服务器程序DeWebServer.exe
3. 双击界面上的dbhello，即可打开在浏览器中显示当前应用



点击“上一项”或“下一项”时，可以显示数据表中不同记录的用户Name。

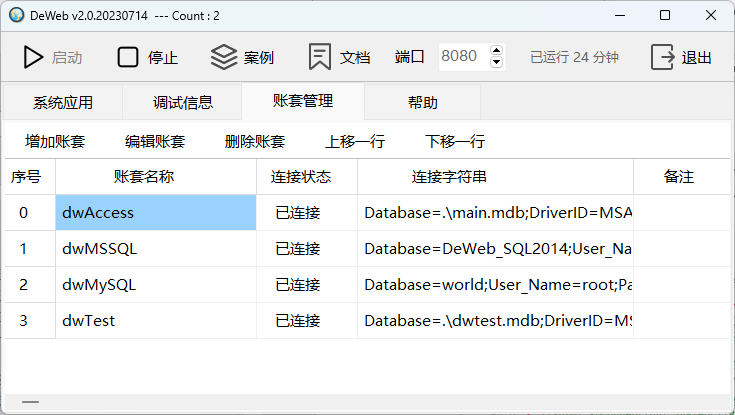
## 4.2 数据库流程

在上述案例中，有些开发者可能只看到Form1的OnShow中打开FDQuery1的代码，不知道如何设置数据库连接。

其实设置数据库连接一般在dpr文件（如dbhello.dpr）的dwLoad函数中，如下：



此时，可以根据序号或名称将DeWebServer的“账套管理”中的账套数据库指定给当前应用。其中，DeWebServer的“账套管理”如图



开发者可以事前先通过“增加账套”、“编辑账套”、“删除账套”、“上移一行”和“下移一行”对当前DeWebServer的账套信息进行管理，以提供给各应用选用。

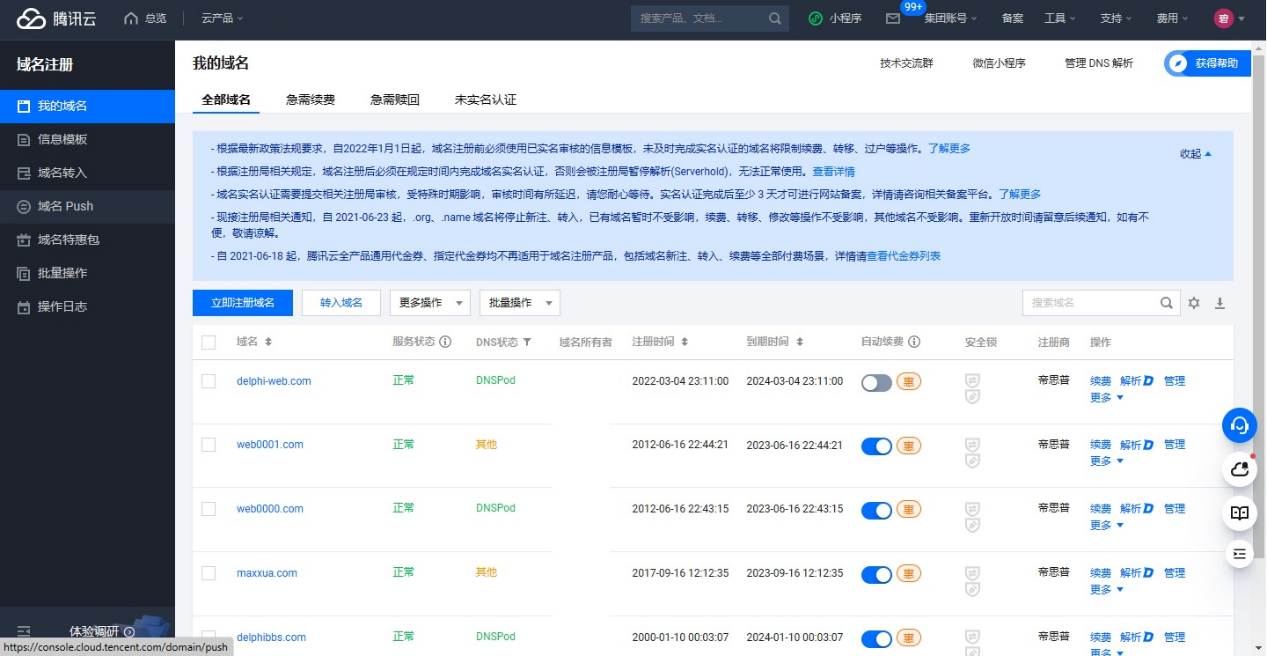
所以，DeWeb的数据库开发流程一般有以下几个步骤：

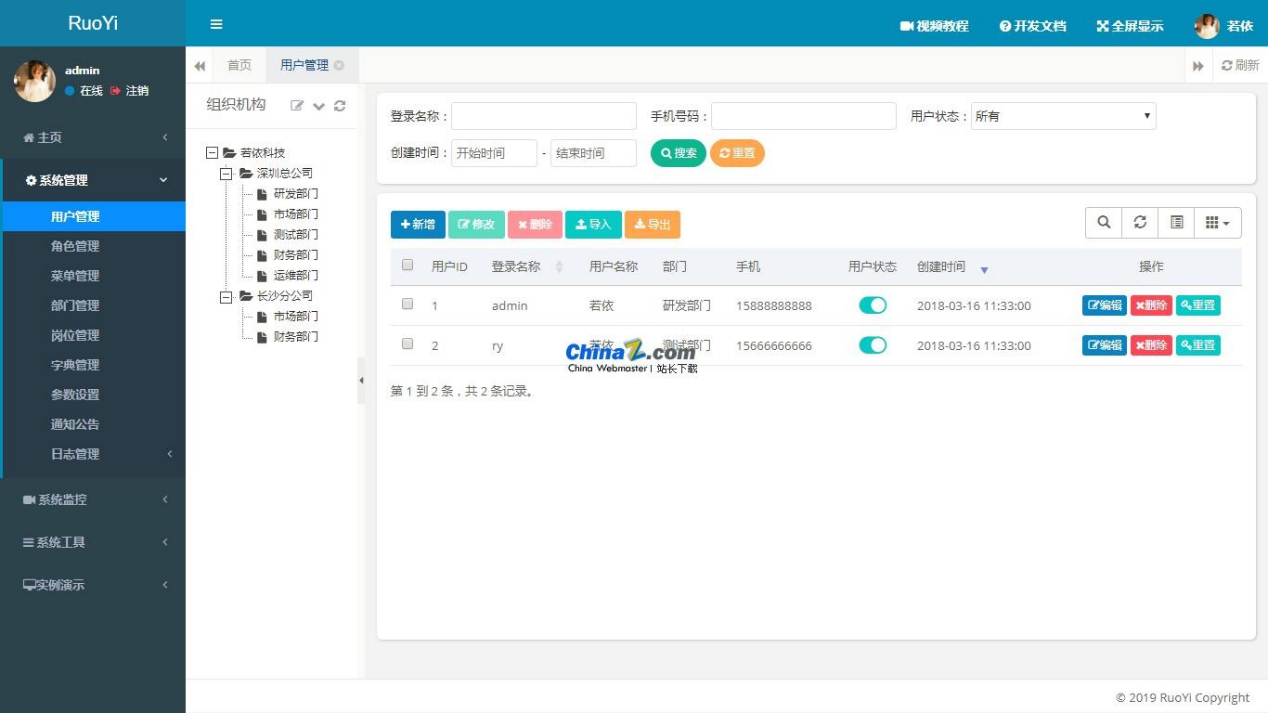
1. 在DeWebServer程序中管理账套信息；
2. 在Form1中放置FireDAC数据库连接控件（如FDConnection1）、相应数据表控件（如FDQuery1、FDTable1等）以及对应的数据库驱动控件（如FDPhysMSAccessDriverLink1、FDPhysMSSQLDriverLink1、FDPhysMySQLDriverLink1等）；
3. 在各工程的dpr文件的**dwLoad**函数中根据序号（dwGetCliHandleByID）或名称（dwGetCliHandleByName）将数据库连接信息赋给Form1中的FireDAC数据库连接（如FDConnection1）;
4. 在Form1的OnShow事件中打开数据表控件（如FDQuery1、FDTable1等），即可进行数据库操作。**注**：不能在OnCreate事件中打开！因为OnCreate时还未指定数据库连接信息！！！

# 多模块开发

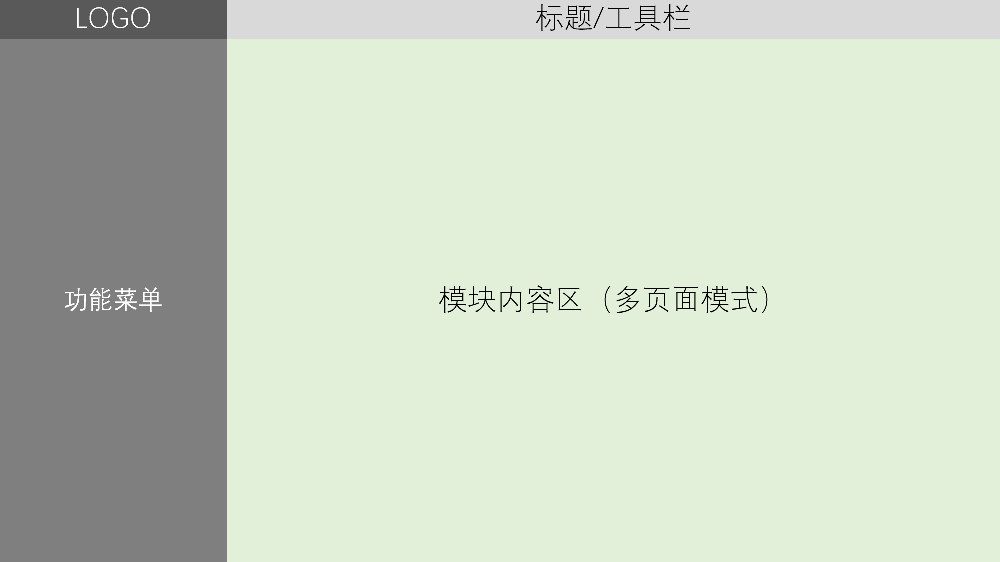
## 5.1 界面框架

在许多网页系统中都能看到类似的界面，如下二图



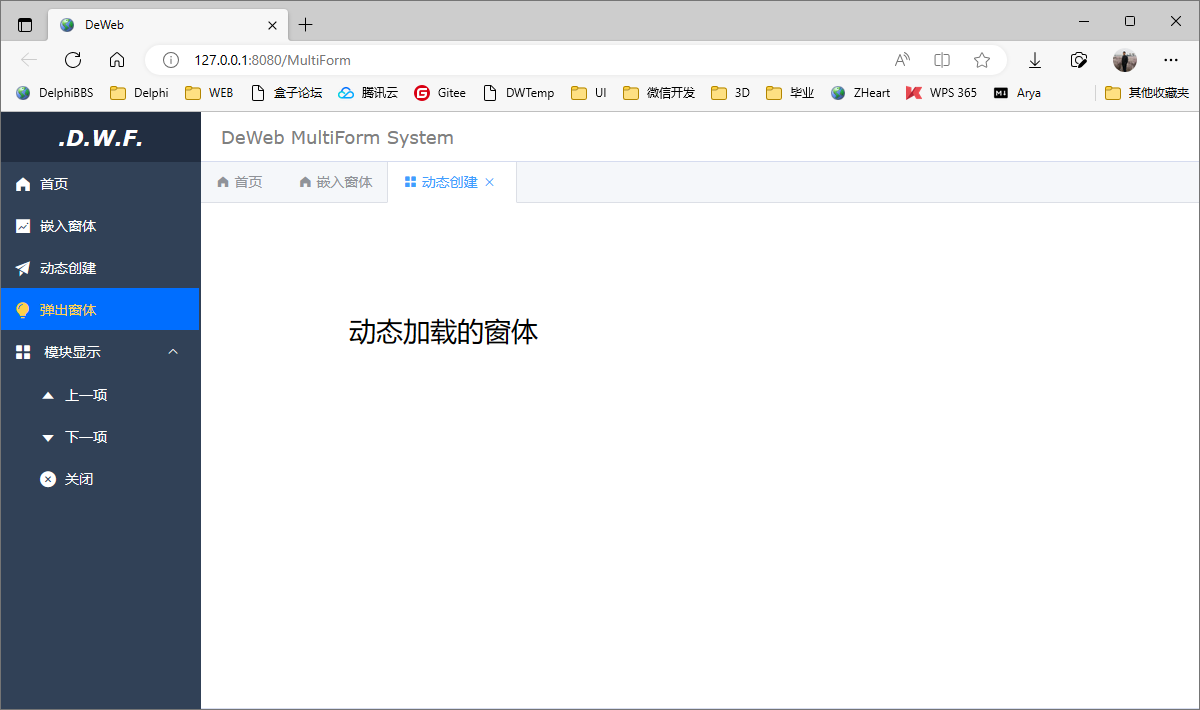


基本界面框架可以简化为：



类似案例可以参考Source\Demos\03 MultiForm\MultiForm.dpr

编译后运行界面如下：



## 5.2 文件组成

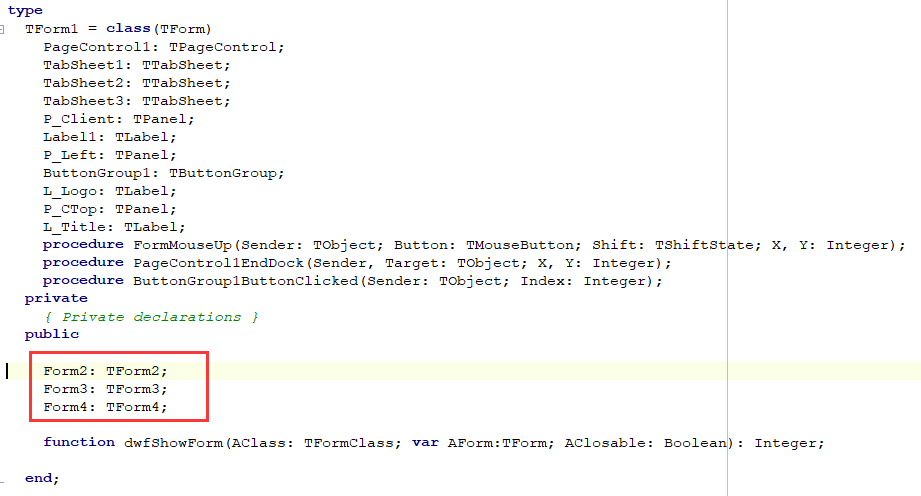
工程主要由工程文件MultiForm.dpr和4个功能单元组成，分别为

* MultiForm.dpr为工程文件，用于承接前端浏览器和后端服务程序DeWebServer；
* unit1 为基础单元，兼显示首页内容；
* unit2 为一个运行默认显示的功能模块（或称子窗体）；
* unit3 为一个通过用户点击动态创建的功能模块；
* unt4 为一个弹出式的功能模块，类似ShowModal；

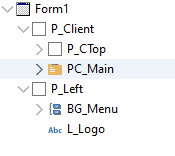
## 5.3 Unit1 基础单元

### 5.3.1 与其他窗体关系

unit1作为当前工程的基础单元，对应的窗体名称固定为Form1（注：这个必须！）。其他的窗体需要放置到Form1的public部分，成为它的子窗体，如下图。



### 5.3.2 界面布局



如图所示：

Form1内主要包括2个Panel, 其中P\_Left位于左侧，align为alLeft;P\_Client位于中部，align为alClient。

P\_Left主要由TLabel控件L\_Logo和TButtonGroup控件BG\_Menu组成。其中L\_Logo用于显示一个文字logo, align为alTop；BG\_Menu用于显示功能菜单项，align为alClient，HelpKeyword为menu（此设置用于使用ButtonGroup来仿垂直菜单显示），事件在OnButtonClicked中。

P\_Client主要由TPanel控件P\_CTop和TPageControl控件PC\_Main组成。其中P\_CTop用于显示标题，如果后期需要，可以放置用户信息、登录、退出等功能按钮；PC\_Main是一个多页面控件，用于显示各个功能模块，具体操作即将其他子窗体（Form2，Form3）放置到对应的TabSheet中。

### 5.3.3 动态创建子窗体

参见unit1中的dwfShowForm函数

### 5.3.4 浏览器分辨率变化事件

当浏览器分辨率发生变化时，会自动激活Form1的OnMouseUp事件。

在本例中

dwSetPCMode(self);

用于置当前为PC模式，也就是一直满屏显示；

iW := PC\_Main.Pages[0].Width;

iH := PC\_Main.Pages[0].Height-10;

用于得到当前PageControl的显示区域尺寸

Form2.Width := iW;

Form2.Height := iH;

if Form3 <> nil then begin

Form3.Width := iW;

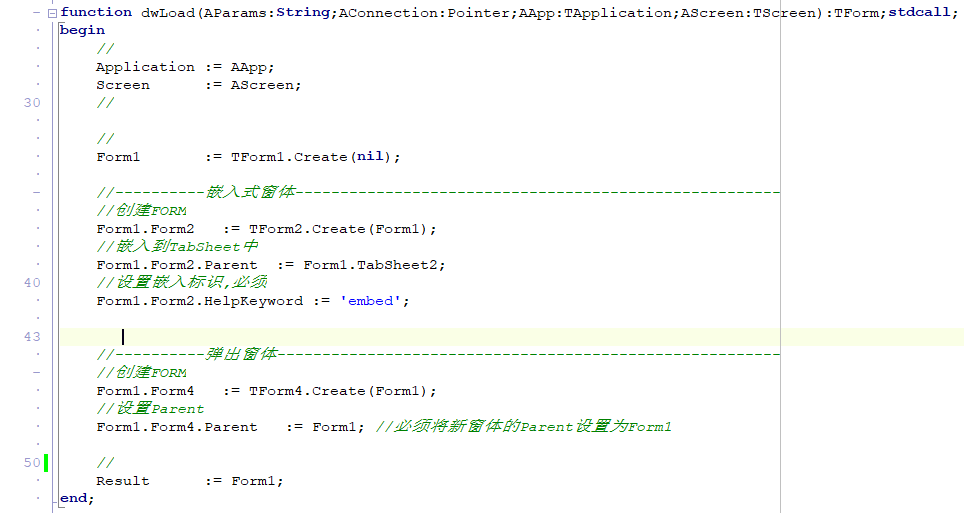
Form3.Height := iH;

end;

用于同步控制Form2和Form3的大小为父区域大小

## 5.4工程文件

MultiForm.dpr为工程文件，最主要的是dwLoad函数。在dwLoad函数中，除了创建主窗体Form1外，还创建了主窗体的子窗体Form2和Form4（Form3在启动完成后动态创建），如下图



注意嵌入式窗体和弹出式窗体的区别主要有2点：

* Parent不同：嵌入式窗体的Parent为对应的TabSheet，弹出式窗体的Parent为Form1;
* HelpKeyword不同：嵌入式窗体的HelpKeyword指定为embed，弹出式窗体的为空值

# 移动端开发

## 6.1 基本原则

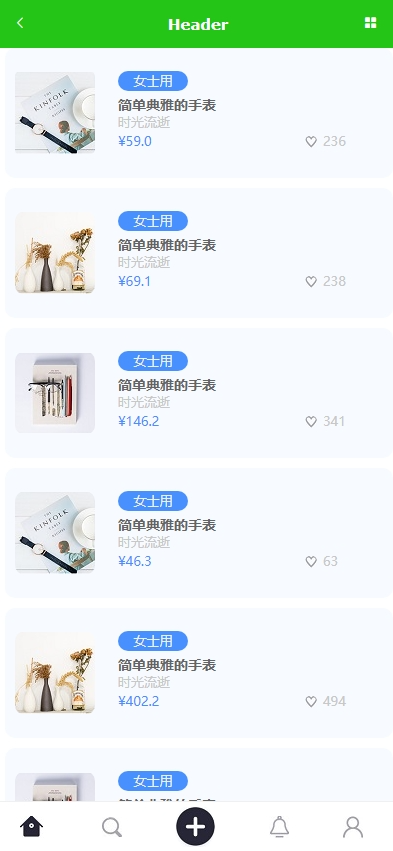
* 和普通竖屏应用一样设计；
* 充分利用Align、AlignWithMargins、Margins、Anchors等特性以适应多种移动端屏幕；
* 在OnMouseUp事件中加入代码

dwSetMobileMode(self,414,736);

并处理可能因屏幕变化造成的显示上的不同。

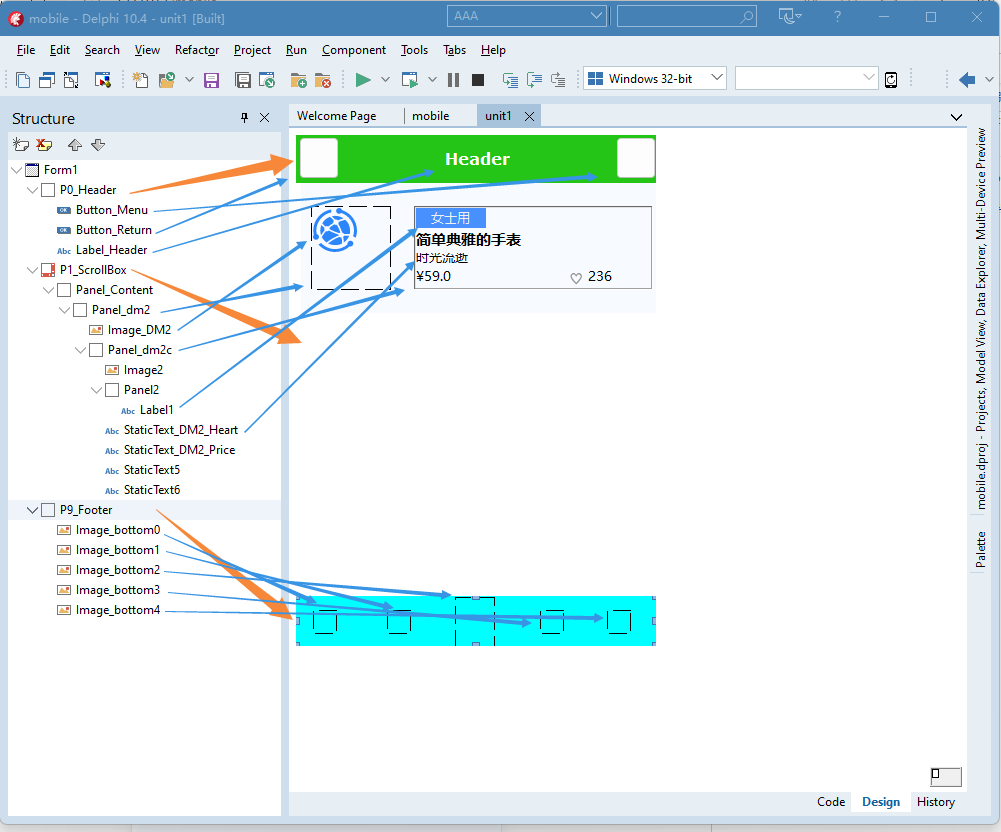
## 6.2 案例分析

以Source\Demos\04 Mobile\mobile.dpr为例，效果图如下：



### 6.2.1 界面布局

界面总体布局如下图



界面主要分上、中、下3个部分，其中：

* 上部为标题栏P0\_Header，类型为TPanel，Align属性为alTop，用于显示标题和功能按钮;
* 中部为内容框P1\_ScrollBox,类型为TScrollBox，Align属性为alClient，用于显示主要内容;
* 下部为按钮栏P9\_Footer，类型为TPanel，Align属性为alBottom，用于显示功能模块按钮。

上部标题栏分左、中、右3个部分，其中：

* 左侧为返回按钮Button\_Return，类型o TButton，Align属性为alLeft，Hint为{"icon":"el-icon-arrow-left", "type": "text"}，分别设置当前按钮的图标和类型；
* 中间为标题Label\_Header,类型为TLabel，Align为alClient；
* 右侧为菜单按钮Button\_Menu，类型o TButton，Align属性为alRight，{"icon":"el-icon-menu","type":"text"}。

中部内容框内仅有一个唯一的子控件Panel\_Content，类型为TPanel，Align为alTop，AutoSize为True，其内部根据需要创建（克隆）多个TPanel以放置拟显示内容；

下部按钮栏内旋转了5个TImage，用作按钮，其Anchors全部设置为[akTop]，相比默认值去除了[akLeft]。这样的Anchors设置用于在设计时水平等距排列后，随着屏幕宽度的变化仍能保持水平等距排列。

**注**：以下按钮仅用作界面演示，未添加具体功能代码。

### 6.2.2 界面处理代码

浏览器分辨率变化事件

当浏览器分辨率变化（用户使用不同分辨率手机打开系统）时会自动激活OnMouseUp事件，一般 在OnMouseUp事件中加入代码

dwSetMobileMode(self,414,736);

一些需要额外处理的也在这里通过代码实现。

# 快速CRUD

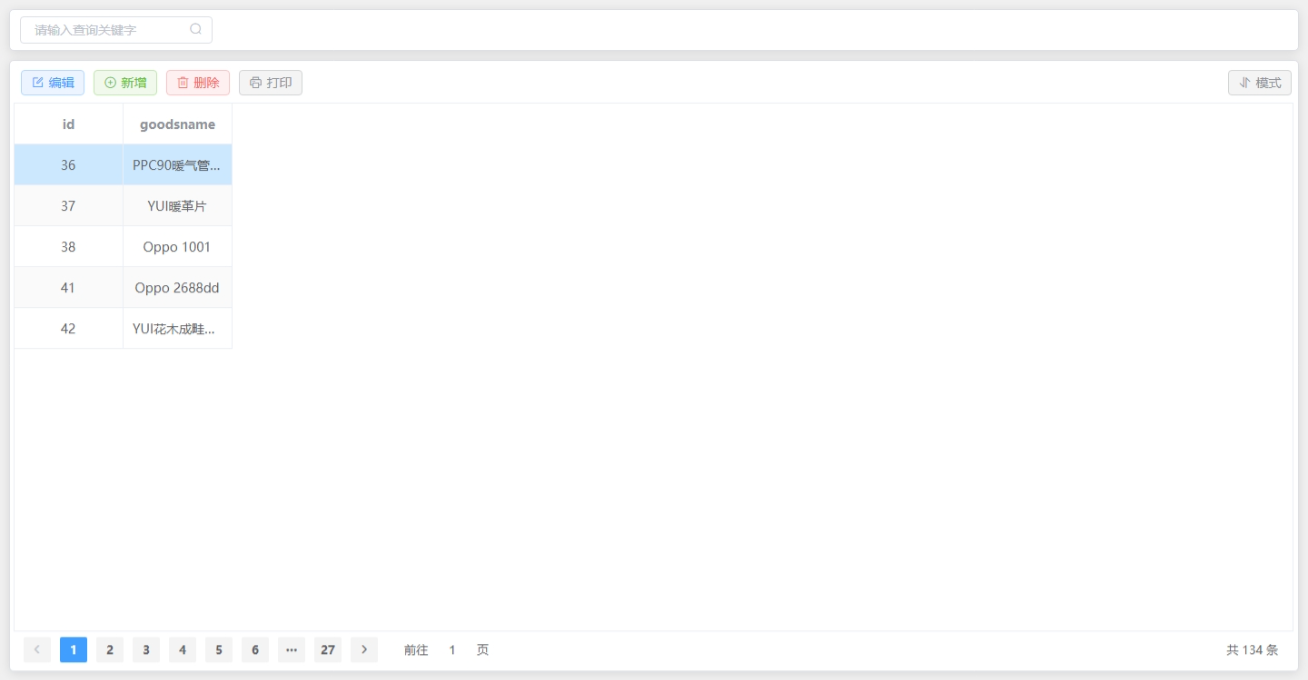
在实际应用中，很多系统需要CRUD功能模块。CRUD即Create(创建)、Retrieve(查询)、Update(更新)、Delete(删除)，也就是日常所说“增删改查”。

在DeWeb开发中，可以通过dwQuickCrud.pas单元(以下称**QuickCrud**)快速创建“增删改查”web功能模块，主要特点有：

* 支持一主多从
* 支持主从表水平或垂直布局；
* 支持智能模糊查询；
* 支持多字段模糊查询；
* 支持自动排序；
* 支持自动分页；
* 支持在线打印；
* 支持新增时自动赋默认值；
* 支持只读属性；
* 支持导出Excel；
* 支持增加自定义按钮。

## 7.1 案例演示

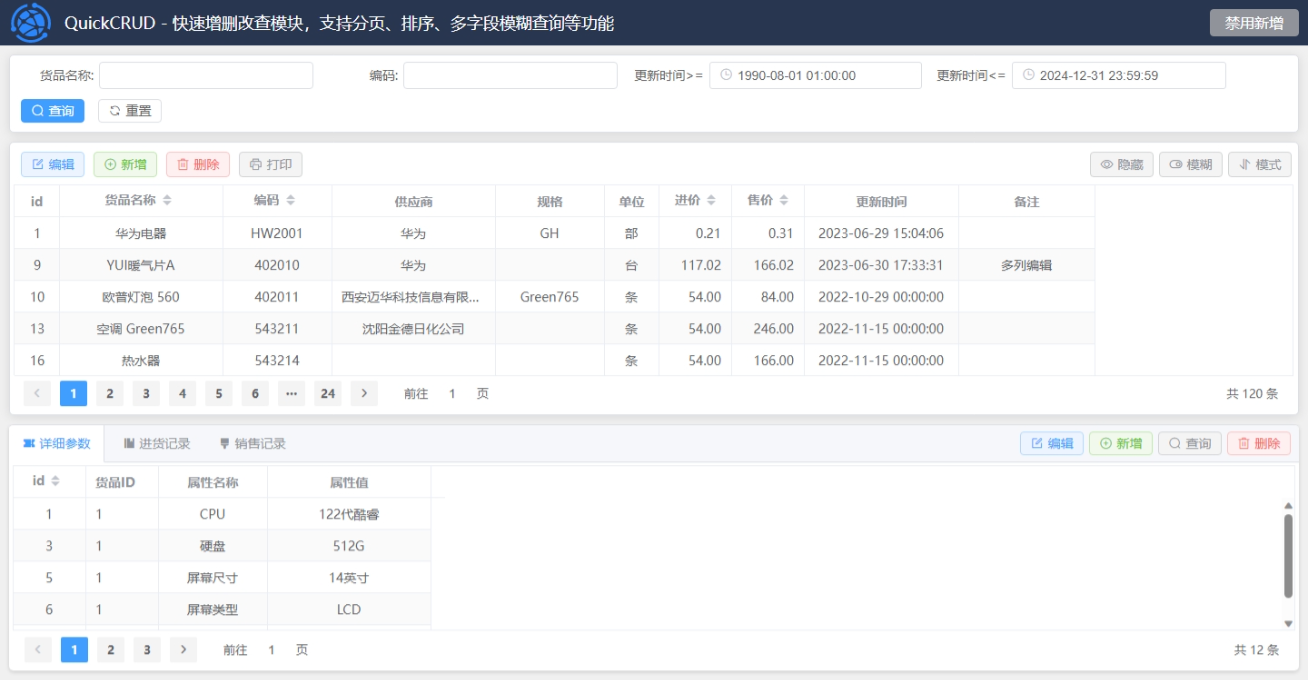
* [QuickCrud入门案例](https://www.delphibbs.com/quickcrudhello)



* [QuickCrud主表案例](https://www.delphibbs.com/quickcrudmaster)



* [QuickCrud一主三从案例](https://www.delphibbs.com/quickcrud)



* [QuickCrud主从水平排列](https://www.delphibbs.com/quickcrudhorz)



## 7.2 QuickCrud入门

（1）复制一个DeWeb的应用，建议hello；

（2）删除Form1的全部控件和事件代码，另存当前工程为quickhellotest.dpr；

（3）在dpr工程文件代码dwLoad函数中增加指定数据库连接代码，如下//<....//>中部分

1. **function** dwLoad(AParams:String;
2. AConnection:Pointer;
3. AApp:TApplication;
4. AScreen:TScreen):
5. TForm;stdcall;
6. **begin**
7. //
8. Application := AApp;
9. Screen      := AScreen;
11. //
12. Form1       := TForm1.Create(**nil**);
14. //<------以下用于为窗体中的FDConnection1指定数据库连接-----------
15. //需要uses引用dwBase, 注意在sharemem后面. 下面有2种方法,可选一种
16. //根据 id (序号，0开始) 获取数据库连接
17. Form1.FDConnection1.SharedCliHandle   := dwGetCliHandleByID(AParams,0);
19. //根据 Name (不区分大小写) 获取数据库连接
20. //Form1.FDConnection1.SharedCliHandle   := dwGetCliHandleByName(AParams,'dwAccess');
21. //>-------------------------------------------------------------------------

24. Result      := Form1;
25. **end**;

（4）在Form1的public中增加变量

1. //QuickCrud 必须的变量
2. qcConfig : **string**;

（5）设置Form1的OnMouseUp事件代码如下

1. **procedure** TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
2. Y: Integer);
3. **begin**
4. //设置当前为PC(电脑端)模式
5. dwSetPCMode(Self);
6. **end**;

（6）修改Form1的StyleName为如下的JSON格式字符串

注意：双引号、冒号、方括号、花括号和逗号必须为半角

1. {
2. "table": "dwf\_goods",
3. "fields":
4. [
5. {
6. "name": "id",
7. "type": "auto"
8. },
9. {
10. "name": "goodsname"
11. }
12. ]
13. }

其中：

* table 表示 当前数据库的表名称；
* fields 表示 当前数据表拟显示的字段数组
* name 表示 当前数据表拟显示字段的字段名称
* type 表示 当前数据表拟显示字段的字段类型，后面有详细说明

（7）设置Form1的OnShow事件代码为

1. **procedure** TForm1.FormShow(Sender: TObject);
2. **begin**
3. //取得QuickCrud的配置，赋给qcConfig变量
4. qcConfig    := StyleName;
6. //自动创建CRUD模块
7. dwCrud(self,FDConnection1,False,'');
8. **end**;

（8）编译运行，启动DWS后，点击quickhellotest即可！界面如下



## 7.3 高级配置

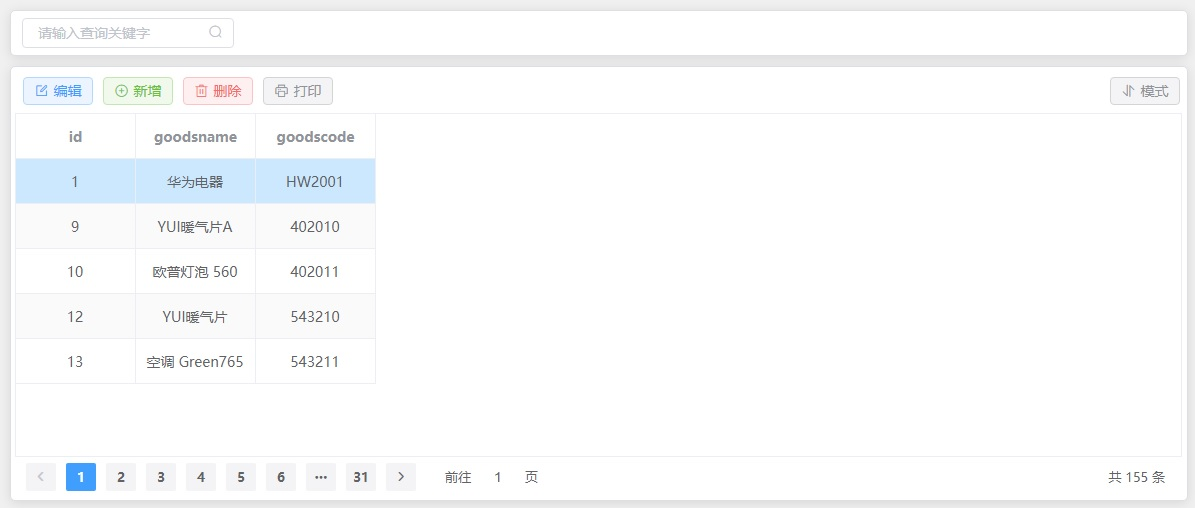
### 7.3.1 配置主表字段

增加主表字段可以通过编辑StyleName属性或编辑qcConfig变量。

**（1）基本配置**

1. {
2. "table": "dwf\_goods",
3. "fields":
4. [
5. {
6. "name": "id",
7. "type": "auto"
8. },
9. {
10. "name": "goodsname"
11. },
12. {
13. "name": "goodscode"
14. }
15. ]
16. }

再编译运行后，界面已增加了字段goodscode，如下图

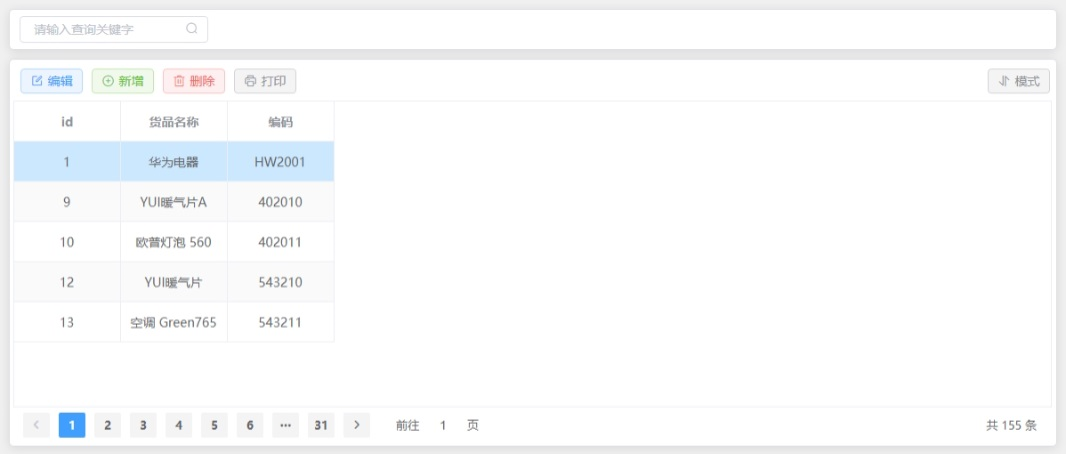


**（2）修改字段显示标题**

修改字段显示标题，可以通过修改StyleName属性的fields中各子节点的caption值，如下：

1. {
2. "table": "dwf\_goods",
3. "fields":
4. [
5. {
6. "name": "id",
7. "type": "auto"
8. },
9. {
10. "name": "goodsname",
11. "caption": "货品名称"
12. },
13. {
14. "name": "goodscode",
15. "caption": "编码"
16. }
17. ]
18. }

再编译运行后，界面已增加了字段的中文标题，如下图

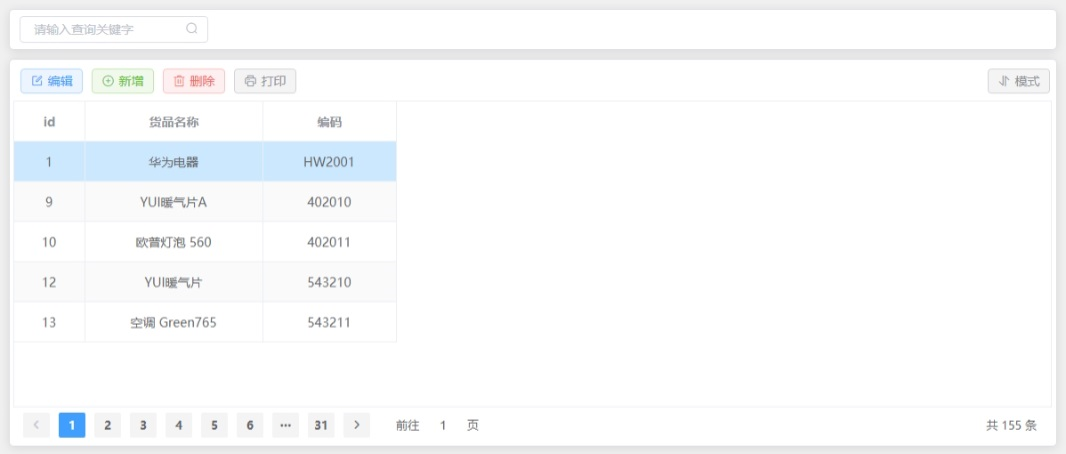


**（3）修改字段显示宽度**

更改字段显示宽度（默认120像素），可以通过修改StyleName属性的fields中各子节点的width值，如下

1. {
2. "table": "dwf\_goods",
3. "fields":
4. [
5. {
6. "name": "id",
7. "type": "auto",
8. "width":80
9. },
10. {
11. "name": "goodsname",
12. "caption": "货品名称",
13. "width":200
14. },
15. {
16. "name": "goodscode",
17. "caption": "编码",
18. "width":150
19. }
20. ]
21. }

再编译已根据设置更新了各字段的显示宽度，效果图如下：



**（4）修改字段对齐方式**

更改字段的对齐方式（默认居中显示），可以通过修改StyleName属性的fields中各子节点的align值，如下

1. {
2. "table": "dwf\_goods",
3. "fields":
4. [
5. {
6. "name": "id",
7. "type": "auto",
8. "width": 80
9. },
10. {
11. "name": "goodsname",
12. "caption": "货品名称",
13. "align":"left",
14. "width": 200
15. },
16. {
17. "name": "goodscode",
18. "caption": "编码",
19. "align":"right",
20. "width": 150
21. }
22. ]
23. }

再编译已根据设置更新了各字段的显示对齐方式，效果图如下：



**（5）修改字段类型**

更改字段的类型，有string, auto, integer, combo, dbcombo, money, date, time, datetime等，默认为string。说明如下：

|  |  |  |
| --- | --- | --- |
| **类型** | **说明** | **备注** |
| string | 字符串类型 | 默认 |
| auto | 自增类型 | 新增和编辑时禁止修改 |
| integer | 整数类型 |  |
| combo | 列表选择类型 | 具体选项由当前节点的list数组决定 |
| dbcombo | 数据库列表选择类型 | 具体选项由当前字段在数据表内值决定 |
| money | 金额类型 | 显示时自动保留2位小数，千分位显示 |
| date | 日期型 | 格式：yyyy-MM-dd |
| time | 时间型 | 格式：hh:mm:ss |
| datetime | 日期时间型 | 格式：yyyy-MM-dd hh:mm:ss |

可以通过修改fields中各子节点的type值，如下：

1. {
2. "table": "dwf\_goods",
3. "fields":
4. [
5. {
6. "name": "id",
7. "caption": "id",
8. "width": 50,
9. "align": "center",
10. "type": "auto"
11. },
12. {
13. "name": "goodsname",
14. "caption": "货品名称",
15. "width": 180
16. },
17. {
18. "name": "goodscode",
19. "caption": "编码",
20. "width": 100
21. },
22. {
23. "name": "provider",
24. "caption": "供应商",
25. "type": "dbcombo",
26. "width": 160
27. },
28. {
29. "name": "spec",
30. "caption": "规格",
31. "width": 120
32. },
33. {
34. "name": "unit",
35. "type": "combo",
36. "caption": "单位",
37. "list":
38. [
39. "",
40. "部",
41. "个",
42. "支",
43. "台",
44. "米",
45. "辆",
46. "套"
47. ],
48. "width": 60
49. },
50. {
51. "name": "inprice",
52. "caption": "进价",
53. "width": 80,
54. "align": "right",
55. "type": "money"
56. },
57. {
58. "name": "price",
59. "caption": "售价",
60. "width": 80,
61. "type": "money",
62. "align": "right"
63. },
64. {
65. "name": "updatetime",
66. "caption": "生产日期",
67. "width": 120,
68. "type": "date"
69. },
70. {
71. "name": "description",
72. "caption": "备注",
73. "width": 120
74. }
75. ]
76. }

再编译已根据设置更新了各字段的类型，效果图如下：



**（6）其他可修改的字段配置**

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **说明** | **示例** | **备注** |
| must | 是否必填 | "must":1 |  |
| readonly | 是否只读 | "readonly":1 |  |
| sort | 是否排序 | "sort":1 |  |
| query | 是否加入多字段查询 | "query":1 |  |
| default | 新增时默认值 | "default":"华为" |  |
| max | 最大值 | "max":999 | 如果是date类型时，"max": "2025-12-31"，其他time/dateime时也类似处理 |
| min | 最小值 | "min":0 | 同max |
| list | 选项值 | "list": ["","部","个","支"] | 仅当combo类型时有效 |

### 7.3.2 配置显示属性

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **说明** | **示例** | **备注** |
| table | 数据表名 | "table":"dw\_member" | 必填 |
| where | 数据表过滤条件 | "where":"id>10" | 默认为空值 |
| pagesize | 每页显示数据行数 | "pagesize":20 | 默认5 |
| rowheight | 数据行高度 | "rowheight":35 | 45 |
| edit | 是否可编辑 | "edit":1 | 默认1 |
| new | 是否可新增 | "new":0 | 默认1 |
| delete | 是否可删除 | "delete":0 | 默认1 |
| print | 是否显示打印按钮 | "print":1 | 默认1 |
| defaultquerymode | 默认查询面板样式 | "defaultquerymode":1 | 0：不显示，1：智能模糊查询；2：多字段查询。默认如果有单字段查询，则为模式2，否则为模式1 |
| hide | 是否显示“隐藏”按钮 | "hide":0 | 仅主从表显示时有效 |
| fields | 当前拟显示的字段数组 |  | 必须 |
| margin | 默认各部分显示边距 | "margin":20 | 默认10 |
| radius | 圆角半径 | "radius":2 | 默认5 |
| editwidth | 编辑/新增面板的宽度 | "editwidth":600 | 默认360 |
| defaulteditmax | 编辑/新增时是否默认最大显示 | "defaulteditmax":1 | 默认0 |
| datawidth | 编辑/新增时每字段宽度 | "datawidth":400 | 默认320 |
| buttoncaption | 是否显示功能按钮标题 | "buttoncaption":2 | 0:全不显示,1:全显示;2:部分显示 |
| slavepagesize | 从表每页显示数据行数 | "slavepagesize":10 | 默认5,仅主从表时有效 |
| slaverowheight | 从表数据行高度 | "slaverowheight":40 | 默认45,仅主从表时有效 |
| mainwidth | 主表显示宽度 | "mainwidth":700 | 如果为主从表,且有此属性,则主从表水平排列,否则垂直排列 |

### 7.3.3 配置从表

（1）先添加slave数组节点

（2）在slave数组中添加若干个从表子节点,结构和主表基本相似

（3）从表的特别设置(主表有的不再描述)

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **说明** | **示例** | **备注** |
| caption | 当前从表页面标题 | "caption" : "销售记录" | 必须 |
| imgeindex | 当前从表页面图标 | "imageindex" : 60 | 见附件 |
| masterfield | 当前从表对应的主表字段 | "masterfield" : "id" | 必须 |
| slavefield | 当前与主表字段对应的字段 | "slavefield" : "gid" | 必须 |

基本配置文件参考如下:

1. {
2. "table" : "dwf\_goods",
3. "fields" : [
4. {
5. "name" : "id",
6. "caption" : "id",
7. "width" : 50,
8. "align" : "center",
9. "type" : "auto"
10. },
11. {
12. "name" : "goodsname",
13. "caption" : "货品名称",
14. "width" : 180,
15. "type" : "string",
16. "must" : 1,
17. "query" : 1,
18. "sort" : 1,
19. "default" : "未命名"
20. },
21. {
22. "name" : "description",
23. "caption" : "备注",
24. "width" : 150
25. }
26. ],
27. "slave" : [
28. {
29. "caption" : "详细参数",
30. "table" : "dwf\_goodsex",
31. "imageindex" : 58,
32. "masterfield" : "id",
33. "slavefield" : "gid",
34. "fields" : [
35. {
36. "name" : "id",
37. "caption" : "id",
38. "width" : 80,
39. "align" : "center",
40. "sort" : 1,
41. "type" : "auto"
42. },
43. {
44. "name" : "gid",
45. "caption" : "货品ID",
46. "align" : "left",
47. "must" : 1,
48. "query" : 1,
49. "type" : "integer",
50. "width" : 80
51. },
52. {
53. "name" : "title",
54. "caption" : "属性名称",
55. "must" : 1,
56. "width" : 120,
57. "query" : 1,
58. "default" : "新属性"
59. },
60. {
61. "name" : "value",
62. "caption" : "属性值",
63. "readonly" : 1,
64. "query" : 1,
65. "width" : 180,
66. "default" : "新值",
67. "must" : 1
68. }
69. ]
70. },
71. {
72. "caption" : "销售记录",
73. "table" : "dwf\_outport",
74. "imageindex" : 60,
75. "masterfield" : "id",
76. "slavefield" : "gid",
77. "fields" : [
78. {
79. "name" : "id",
80. "caption" : "id",
81. "width" : 50,
82. "align" : "center",
83. "type" : "auto"
84. },
85. {
86. "name" : "gid",
87. "caption" : "货品ID",
88. "type" : "integer",
89. "width" : 80
90. },
91. {
92. "name" : "operator",
93. "caption" : "经办人",
94. "query" : 1,
95. "width" : 120,
96. "must" : 1
97. }
98. ]
99. }
100. ]
101. }

## 7.4 常见问题

本节由热心网友“**寒酥**”友情提供！特别感谢！

### 7.4.1 添加自定义按钮

在窗体上创建一个按钮，命名随意（Button\_ToXls）；

然后再窗体的onshow事件中新增如下代码即可。

注:

（1）在OnShow事件中需要将新增按钮的代码写在创建quickcrud代码后面。

1. qcAdd(Button\_ToXls);

### 7.4.2读取本地JSON文件

目前QuickCrud中JSON字符串存放在StyleName中，每次更新及修改比较麻烦，且需要打开delphi。为解决此问题，创建一个JSON文件，存放目录自定义，目录内存放各种JSON配置文件，就相当于把StyleName中内容放到本地JSON文件中，需要修改、添加可以直接修改此JSON文件即可，另外若是设置了JSON文件，系统无法读取出来请检查文件格式。以下为代码示例：

1. qcConfig:=TFile.ReadAllText('TableJson/v\_df2\_ydbf\_lpzf.json');

### 7.4.3自定义时间查询条件

自定义时间查询条件，如默认当前时间，默认本月、本年度、上年度等，在formcreate事件中写如下代码

1. **procedure** TForm\_TJFX1.FormCreate(Sender: TObject);
2. **var**
3. monthFirstDay,yearFirstDay,LastMonthFirsyDay,LastMonthLastyDay:**string**;
4. **begin**
5. // 定义本年度第一天，本月度第一天，上月第一天与最后一天
6. monthFirstDay:=FormatDateTime('yyyy-mm-01',Now);   //本月第一天
7. yearFirstDay:=FormatDateTime('yyyy-01-01',Now);    //本年度第一天
8. LastMonthFirsyDay:=FormatDateTime('yyyy-mm-1',IncMonth(Now,-1)); //上月第一天
9. LastMonthLastyDay:= DateToStr(startofthemonth(Now)-1); //上月最后一天 ,需引用System.DateUtils
11. //自动读取JSON内容
12. qcConfig:=TFile.ReadAllText('TableJson/v\_df2\_ydbf\_lpzf.json');
14. qcConfig := StringReplace(qcConfig, '"1990-01-01"', yearFirstDay, [rfReplaceAll]);
15. **end**;

此代码需要先在JSON文件中创建需要替换的对象的字符串，如1990-01-01，此方法同样适用于其他字段动态创建。

### 7.4.4获取主从表中选中行方法

（1）获取主表中选中行方法，以下示例为点击按钮时获取主表选中行并取字段值的方法：

1. **procedure** TForm\_datain.btn\_editClick(Sender: TObject);
2. **var**
3. oFDQuery    : TFDQuery;
4. oStringGrid : TStringGrid;
5. sName       : **string**;
6. SelctRow    : Integer;    //   定义选中行的下标
7. **begin**
8. //  获取当前fdquery以及stringgrid，0为主表
9. oFDQuery    := qcGetQuickCrudFDQuery(self,0);
10. oStringGrid := qcGetQuickCrudStringGrid(Self,0);
11. //  获取选中行的下标
12. SelctRow:=oStringGrid.Row;
13. //取选中行的值 ，选中行的的值RecoNo就等选中行的下标
14. oFDQuery.RecNo := SelctRow;
15. lbl\_id.Caption := oFDQuery.FieldByName('id').AsString;
16. **end**;

（2）获取从表中当前选中行方法，以下示例为点击按钮时获取第一个从表选中行并取字段值导出数据的方法：

1. **procedure** TForm\_tjfx2.Button\_ToXlsClick(Sender: TObject);
2. **var**
3. sSQL    : **string**;
4. oQuery  : TFDQuery;
5. sDir    : String;
6. sFile   : String;
7. JZND    : String;
8. SlaveIndex:Integer;
9. oPC     :TPageControl;
10. **begin**
11. // 动态获取当前激活的从表的下标，因为获取的下标从0开始，所以需要加1
12. oPC     := TPageControl(FindComponent('PC\_Slave'));
13. // 因为从表从1开始，获取的下标从0开始，所以需要加1
14. SlaveIndex:=oPC.ActivePageIndex+1;
16. //取从表的SQL语句，主表为0，从表第一个为1，第二个为2，以此类推。
17. sSQL    := qcGetQuickCrudSQL(Self,SlaveIndex);
18. **try**
19. //打开Query
20. oQuery  := TFDQuery.Create(self);
21. oQuery.Connection   := Form1.FDConnection1;
22. oQuery.SQL.Text     := sSQL;
23. oQuery.FetchOptions.RecsMax := 999999;
24. oQuery.Open;
26. //  取就诊年度字段用于导出时标题
27. JZND := oQuery.FieldByName('就诊年度').AsString;
28. //转成EXCEl
29. sDir    := ExtractFilePath(Application.ExeName);
30. sFile   := 'download/'+JZND+'年度测算明细\_'+FormatDateTime('MMDD\_hhmmsszzz\_',Now)+IntToStr(Random(1000))+'.xls';
31. dwExportToXls(sDir+sFile,oQuery);
32. //打开下载
33. dwOpenUrl(self,'/'+sFile,'\_blank');
34. **finally**
35. oQuery.Free;
37. **end**;
38. **end**;

# 看板（KANBAN）管理开发

## 8.1 什么是看板（KANBAN）?

看板（KANBAN）- 不可忽视的协作工具

看板是什么？一块可以看的板子？你要是这么想的话也不能算你错...但是，看板在日语里面最早的解释是招牌，向客户传递信息，招揽生意。“可以看的板”仅仅是看板的一种形态，可以说这个日语词汇的本来意思就不是冲着“可以看的板”去的。看板的本意还具有信息传递的意思，更有很大程度的商业的意味，而不限制于形式。

而看板管理，就来自日语“看板”，是丰田生产模式中的重要概念，指为了达到准时生产方式（JIT）控制现场生产流程的工具。准时生产方式中的拉式（Pull）生产系统可以使信息的流程缩短，并配合定量、固定装货容器等方式，而使生产过程中的物料流动顺畅。

DeWeb开发的一个典型看板管理界面如下：



演示网址：<https://www.delphibbs.com/bdv001>

## 8.2 开发方法

看板管理应用开发和普通开发没有区别，只不过使用了特殊的颜色和控件。

下面，以bdv001为例，介绍开发方法

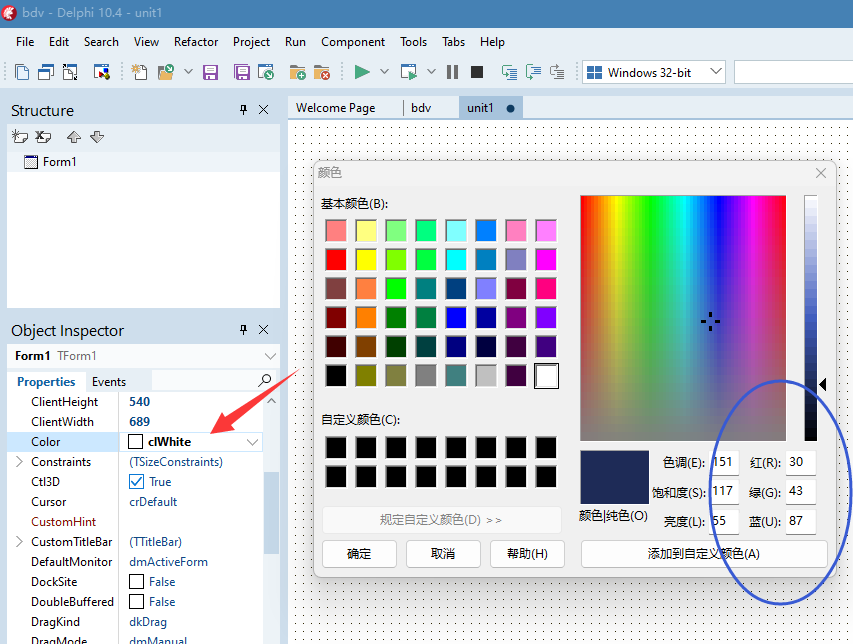
### 8.1.1 新建工程

新建工程，可以在Source\Demos创建一个新目录bdv；然后，创建一个DLL工程，配置选项。这个过程的操作相对较多，建议直接复制Source\Demos\01 Hello目录，粘贴到Source\Demos目录，然后重命名目录名称为bdv。

用Delphi打开Source\Demos\bdv\Hello.dpr， 然后菜单File -> Save Project As , 保存为bdv.dpr.

### 8.1.2 基本设置

1. 删除原来所有的控件和代码；
2. 设置背景色。选一个合适的颜色，一般参考几个案例就可以。如图



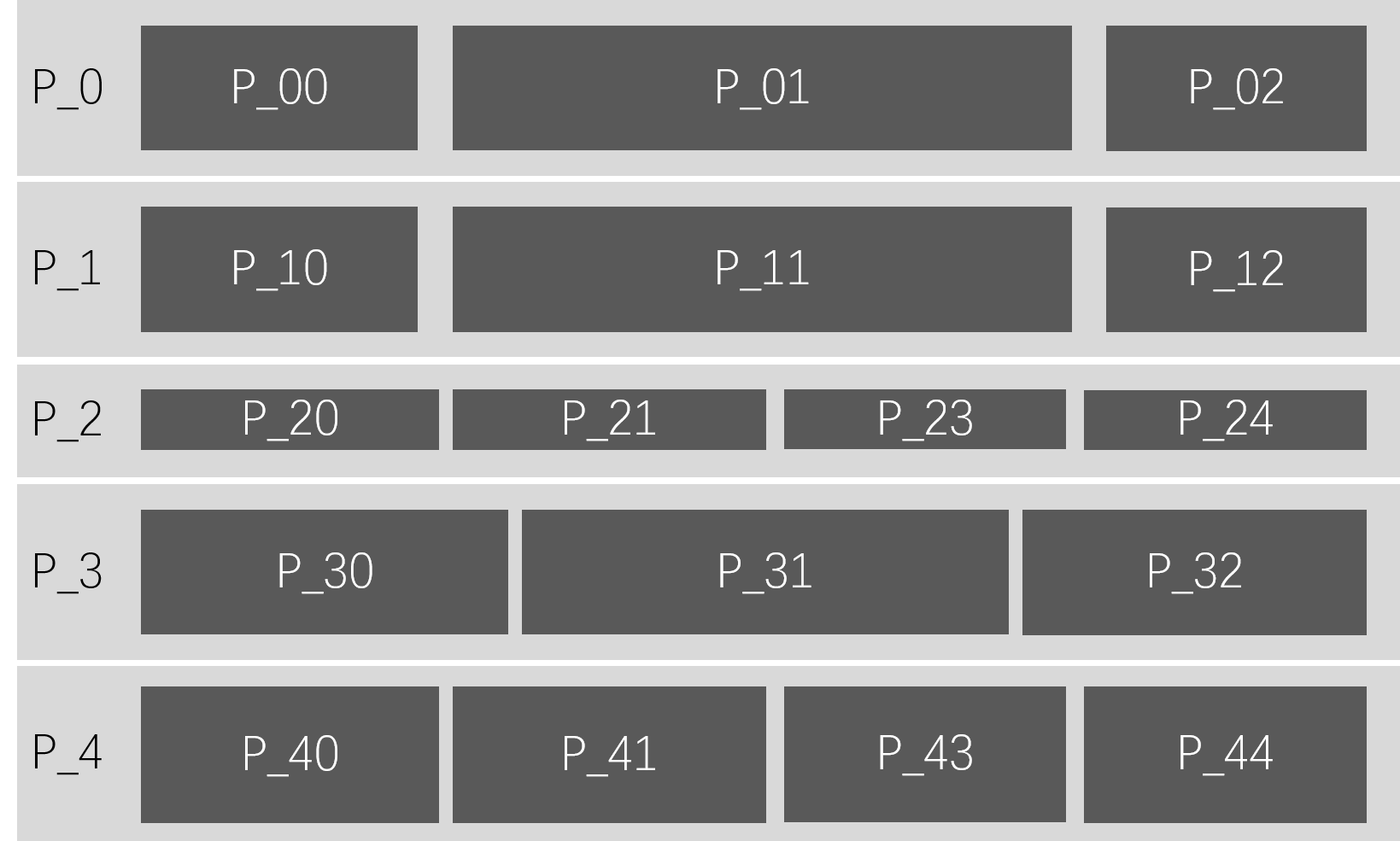
1. 设置电脑模式

在OnMouseUp中增加代码

1. **procedure** TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton;
2. Shift: TShiftState; X, Y: Integer);
3. **begin**
4. dwSetPCMode(Self);
5. **end**;

### 8.1.3 总体布局

根据对界面的观察，可以分为垂直排列的5个Panel，分别为P\_0, P\_1, P\_2, P\_3, P\_4，每个Panel内有3或4个子Panel, 布局基本如下：



### 8.1.4 显示分类

根据对拟显示数据分析，显示控件可以分为以下几类：

1. 标签类。如：



图1 普通文本

该类显示直接用TLabel来实现，根据需要改变Caption, Align, Alignment,Font.Name, Font.Color,Font.Size,Font.Style等属性。

一般情况下其Color属性无效，如果需要使用Color属性，可以设置其Transparent为False,也可以设置其父控件的Color。



图2 多行文本

该类显示的实现方法和图1的基本一样，只需要在Caption中加入换行即可。一般情况下Caption中不支持换行，可以通过安装Raize套件，可以使Caption支持换行。

注意：还是使用TLabel控件实现，不使用Raize套件中的控件，只使用Raize中的属性编辑器。



图3 组合文本

该类显示通过使用多个TLabel控件实现。

1. 边框类。如：



图4 图形边框

该类显示的边框比较特别，所以采用了一个图片作为底图（这个本案例中唯一使用的图片）。为了更好地进行布局，本例中使用了一个TPanel为父控件，里面依次放入TImage和TLabel,这2个控件都设置Align为alClient。如果TLabel没有显示在图片中方框的中间，可以通过AlignWithMargins和Margins属性进行微调。



图5 标准边框

该类显示可以通过控件的Hint属性中的dwstyle项来配置，如Hint为

1. {"dwstyle": "border-bottom:solid 1px #178BA5;"}



图6 特殊边框一

该类显示可以通过使用Panel\_\_dvbox（即使用TPanel控件，其HelpKeyword设置为dvbox）控件来解决，参考“10 控件使用”相关内容。其HelpContext设置为12。



图7 特殊边框二

该类显示可以通过使用Panel\_\_dvbox（即使用TPanel控件，其HelpKeyword设置为dvbox）控件来解决，参考“10 控件使用”相关内容。其HelpContext设置为13。



图8 特殊边框三

该类显示可以通过使用Panel\_\_dvbox（即使用TPanel控件，其HelpKeyword设置为dvbox）控件来解决，参考“10 控件使用”相关内容。其HelpContext设置为10。

1. 图表类。



该类显示可以通过使用Memo\_\_echarts（即使用TMemo控件，其HelpKeyword设置为echarts）控件来解决，参考“10 控件使用”相关内容。以下都是。

图9 环图



图10 玫瑰图



图11 雷达图

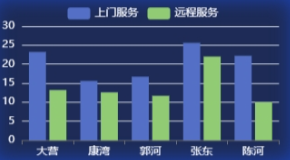


图12 柱图（垂直）



图13 柱图（水平）

1. 表格类。



图14 表格图

该类显示可以使用StringGrid来实现，也可以通过多个Panel来组合实现。

## 8.3 小技巧

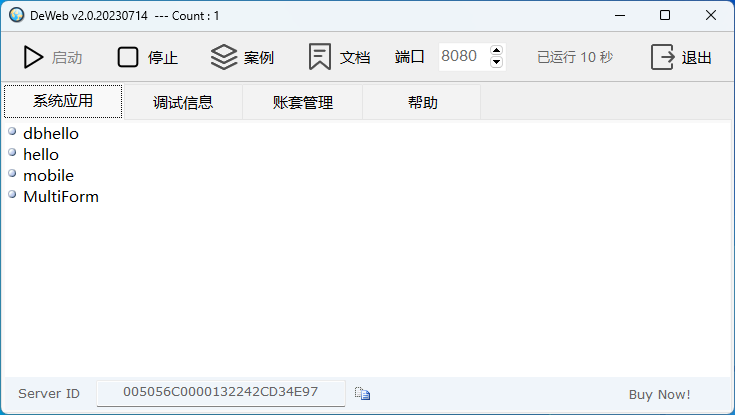
### 8.3.1 全屏显示

### 8.3.2 自动缩放

# 服务程序

## 9.1 DeWebServer使用

### 9.1.1 主界面



其中，

* 按钮“启动”用于启动DeWeb服务；
* 按钮“停止”用于停止DeWeb服务；
* 按钮“案例”用于打开DeWeb案例网站；
* 按钮“文档”用于打开DeWeb文档网站；
* “端口”用于显示和修改端口号（先“停止”后方可修改）；
* 按钮“退出”用于退出服务；
* 中间主显示区
  + 系统应用，用于显示当前系统中的应用，双击可自动打开；
  + 调试信息，用于显示当前系统的调试信息；
  + 账套管理，用于管理当前系统的数据为账套；
  + 帮助
* 底部ServerID,与当前计算机硬件相关，用于获取授权；

### 9.1.2 命令行启动

* 带参数命令行运行

如果1个参数，则为运行端口号，如：DeWebServer 8081

如果3个参数，则分别为端口号、窗体Left、窗体Top

如果5个参数，则分别为端口号、窗体Left、窗体Top、窗体Width、窗体Height

## 9.2 配置文件

DeWebServer的配置文件为同目录下的config.json文件，基本如下

1. {
2. "port": 8080,
3. "resource": 0,
4. "third": "http://cdn.delphibbs.com",
5. "redir": [],
6. "upload\_dir": "upload",
7. "max\_upload\_size": 80,
8. "errorhtml": "<html><body><br/><p style='text-align:center'>!!!deweb web not exist !</p></body></html>",
9. "disconnect": "Network is disconnected!",
10. "nightrestart": 0,
11. "database":
12. [
13. {
14. "name": "dwAccess",
15. "string": "Database=.\\main.mdb;DriverID=MSAcc",
16. "remark": "",
17. "params": []
18. },
19. {
20. "name": "dwMSSQL",
21. "string": "Database=DeWeb\_SQL2014;User\_Name=sa;Password={deweb@12345};Server=www.delphibbs.com;DriverID=MSSQL",
22. "remark": "",
23. "params": []
24. },
25. {
26. "name": "dwMySQL",
27. "string": "Database=world;User\_Name=root;Password={deweb@12345};Server=www.delphibbs.com;DriverID=MySQL",
28. "remark": "",
29. "params":
30. [
31. "CharacterSet=gb2312"
32. ]
33. },
34. {
35. "name": "dwTest",
36. "string": "Database=.\\dwtest.mdb;DriverID=MSAcc",
37. "remark": "",
38. "params": []
39. }
40. ]
41. }

其中：

**port** : 整型，为默认启动时的端口；

**resource** : 整型，为启动时资源文件的位置。0表示本地源，1表示公用源，2表示第三方源，暂时未使用；

**third** : 字符串型，和resource为2时配合使用，为第三方源网址，暂时未使用；

**redir** : 字符串二维数组，为自动地址切换表，暂时未使用；

**upload\_dir**:为上传文件存放目录；

**max\_upload\_size**为上传文件最大size，单位为MB

**errorhtml** : 字符串型，为应用不存在时的提示（HTML格式）。

**database**: JSON对象数据组，为数据库账套信息。

**disconnect** : 字符串型，为网络断线时的提示字符串。

# 控件使用

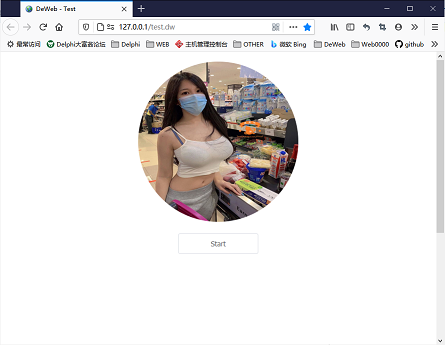
## Form

* 目前主窗体Name应为Form1，其他窗体Name应以"form"开头，

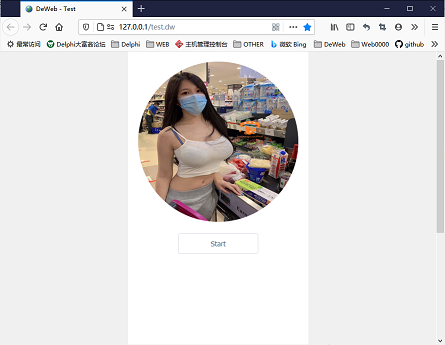
如Form2,Form3,Form\_VIP,...

* 有2种模式：

1. 如果TransparentColor=False，则为全色模式，如图：



1. 如果TransparentColor=True，则为双色模式，如图：



此时，TransparentColorValue为网页底色，Color为主体部分颜色

* 属性VertScrollBar.Visible决定当窗体高度不够时是否自动显示滚动条
* OnCreate代码有效
* OnShow代码有效

在OnShow中可以通过dwGetProp(Self,'params')得到URL中的参数

* 如果把工程名改为default.dpr，则可以URL中不输入http://127.0.0.1:8080/default, 直接通过<http://127.0.0.1:8080>即可访问
* ip可以通过dwGetProp(Self,'ip')得到（目前好像不准确）；
* AlphaBlend=True时可以通过DeWebserver.exe的Html按钮生成纯html文件，可用于非DeWeb环境；
* 在代码中设置

DockSite :=True;

可以在动态创建和删除控件时刷新整个窗体；

* 多平台适配代码在OnMouseUp中。

X,Y分别表示当前设备的Width/Height；

Button : mbLeft : 屏幕纵向, mbRight:屏幕横向；

Shift：ssShift, ssAlt, ssCtrl,ssLeft, ssRight,

分别对应0:未知/1:PC/2:Android/3:iPhone/4:Tablet

另外，浏览窗体的

screenWidth可以通过dwGetProp(Self,'screenwidth')得到；

screenHeight可以通过dwGetProp(Self,'screenheight')得到；

innerWidth可以通过dwGetProp(Self,'innerwidth')得到；

innerHeight可以通过dwGetProp(Self,'innerheight')得到；

clientWidth可以通过dwGetProp(Self,'clientwidth')得到；

clientHeight可以通过dwGetProp(Self,'clientheight')得到；

物理分辨率宽度可以通过dwGetProp(Self,'truewidth')得到；

物理分辨率高度可以通过dwGetProp(Self,'trueheight')得到；

设置像素比可以通过dwGetProp(Self, 'devicepixelratio')得到；

其中：

screenWidth/screenHeight为屏幕分辨率。 注意：移动端为虚拟屏幕分辨率

innerWidth/innerHeight为可视区的宽高，包括了滚动条的宽度

clientWidth/clientHeight为可视区的宽高，不包括了滚动条的宽度

* 页面滚动事件代码在OnHelp中。

Data 表示当前滚屏的顶部，例如：设置

Label1.Top := Data；

可以将当前Label1一直在顶部

如果想一直在底部, 需要取得浏览器高度, 需要根据设备是移动端还是PC端分别进行处理。

如果是PC端， 则需要用到clientwidth/clientheight

如果是移动端， 则需要用到screenwidth/screenheight

* 关于多窗体，如何ShowModal，需要先把Form2变量定义Form1的public中，然后在dpr中中手动创建

Form1.Form2 := TForm2.Create(Form1);

Form1.Form2.Parent := Form1;

此窗体内的按钮关闭ShowModal的窗体可以用类似以下代码

dwRunJS('this.'+self.Name+'\_\_vis=false;',self);

* 如果需要多个FORM显示在首页的PageControl的多个TabSheet中，可以如下设置：

将Form1,Form2,Form2,...定义到Form1的public中

**注：**

* 不要使用全局变量
* 多窗体系统需要把所有非主窗体作为Form1的子对象，放到private或public中；其他窗体以Form1为Owner创建,也就是类似      
   Form1.Form2   := TForm2.Create(Form1);//括号内应为主窗体的名称
* 在程序中不能直接使用窗体变量(如：Form1,Form2,...)进行操作，必须采用self来操作(除以上的创建外)；
* 窗体操作操作本单元的代码用self代替；
* 如果Form2需要操作Form3, 可以用如

TForm1(self.Owner).Form3.Label3.Caption := 'new';

* 公共函数库需要用来窗体，建议把窗体作为一个输入参数

## Button

* 响应OnClick事件
* 响应OnEnter事件
* 响应OnExit事件
* 设置ElevationRequired为True时，点击一次时会自动禁用该按钮，以防止多次点击。可以在按钮事件加入类似代码以解除禁用

Button1.ParentCustomHint := False;

或者以下

dwRunJS('this.'+dwPrefix(Button1)+Button1.Name+'\_\_dis=false;',self);

以上2种代码使用一种即可，其中第2种方法更高效一些。

* 设计阶段通过更改Hint来进行配置
* 改变按钮样式(默认为空或无type)

分别为

{"type":"primary"}

{"type":"success"}

{"type":"info"}

{"type":"warning"}

{"type":"danger"}

{"type":"text"}

* 自定义颜色

{"backgroundcolor":"rgb(51,123,208)" }

其中：backgroundcolor为按钮背景色

* 设计阶段通过更改Hint来改变样式(默认为空)

分别为

{"style":"plain"}

{"style":"round"}

{"style":"circle"}

* 设计阶段通过更改Hint来改变图标(默认为无)

例：

{"icon":" el-icon-star-off"}

右侧显示图标用（如果此项创建时没有设置，后面不能再设置了）

{"righticon":" el-icon-star-off"}

各图标名称见附件或<https://element.eleme.cn/#/zh-CN/component/icon>

* 通过更改Hint来改变圆角半径

{"radius":"5px"}

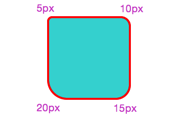
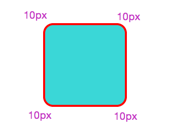
{"radius":"5px 5px 0px 0px"}

数值有三种表示方法：px、%、em，一般推荐在用px；

如果你在 border-radius 属性中只指定一个值，那么将生成 4 个 圆角。

但是，如果你要在四个角上一一指定，可以使用以下规则：

* 四个值: 第一个值为左上角，第二个值为右上角，第三个值为右下角，第四个值为左下角。
* 三个值: 第一个值为左上角, 第二个值为右上角和左下角，第三个值为右下角
* 两个值: 第一个值为左上角与右下角，第二个值为右上角与左下角
* 一个值： 四个圆角值相同



* 注：按钮仅有图标，没有文字标题时，可以通过设置Cancal为True, 这样可以使图标居中。缺点是不能动态设置文字标题。
* 常用的左侧有圆角，右侧没有，则可以

{"radius":"10px 0 0 10px"}

如图



* 左侧无圆角，右侧有，则可以类似设置hint

{"radius":"0 10px 10px 0"}

如图



* 注意图标和颜色可以同时配合使用，例：{"type":"success","icon":" el-icon-star-off"}
* **注意**：除双引号内的部分，不要使用全角符号！，请特别注意引号和逗号
* 本地执行javascript代码（**注：本功能需要熟练掌握javascript知识，请谨慎使用！**）

可以在Hint中添加可在本地执行的javascript代码，执行时机分别o为OnEnter,OnExit,OnClick

类似：

{"onenter":"this.Button1\_\_cap='enter';","onexit":"this.Button1\_\_ cap ='exit';"}

表示当鼠标进入时，置Button1的caption为enter；

当鼠标退出时，置Button1的caption为exit；

或者

{"onclick":"this. Button1\_\_cap='click';"}

表示当鼠标单击时，置Button1的caption为click；

在Hint中如下设置，可以在点击时启动进度条

{"onclick":"this.dwloading=true;"}

可以控制的变量有：

Button1\_\_top : 上边距

Button1\_\_lef : 左边距

Button1\_\_wid : 宽

Button1\_\_hei : 高

Button1\_\_vis : 可见性

Button1\_\_cap : 标题

其中Button1需要根据当前控件名相应更改，支持在控件A中控制另一控件B的属性。



## Calendar

* 支持OnClick事件
* 不支持OnChange事件

## CheckBox

## ComboBox

* 高度可以通过设置Hint为{"height":35}来实现；
* 高度也可以通过改变字体来实现。但实际显示效果中字体还是默认的，仅高度改变
* 指定ItemIndex时，需要手动指定Text，类似：

ComboBox1.ItemIndex := 1;

ComboBox1.Text := ComboBox1.Items[ComboBox1.ItemIndex];

* 取得ItemIndex时，需要手动取得，类似：

procedure TForm1.Button\_GetRowClick(Sender: TObject);

var

I : Integer;

iRes : Integer;

begin

//

iRes := -1;

for I:=0 to ComboBox1.Items.Count-1 do begin

if ComboBox1.Items[I] = ComboBox1.Text then begin

iRes := I;

break;

end;

end;

//

ComboBox1.ItemIndex := iRes;

//

dwShowMessage('ItemIndex : '+IntToStr(ComboBox1.ItemIndex),self);

end;

## DateTimePicker

## Edit

* 支持OnChange事件
* 支持OnEnter事件
* 支持OnExit事件
* 支持OnMouseEnter事件
* 支持OnMouseLeave事件
* **注意**：
* Edit的Text值中有“或<或>时会影响解析，需要用\"或\<或\>来代替；同时在代码中注意：处理时多一个\，而在前端显示时无此\
* Edit的Text值前后的空格会被忽略
* 可以通过设置Hint来前置图标，如{"prefix-icon":"el-icon-s-home"}
* 可以通过设置Hint来后置图标，如{"suffix-icon":"el-icon-s-home"}
* 可以通过设置Hint来设置输入提示，如{"placeholder":"请输入用户名"}
* 可以组合，如：{"placeholder":"请输入用户名","prefix-icon":"el-icon-s-home","suffix-icon":"el-icon-search"}
* 为了保持输入样式的一致性，在Delphi的IDE中修改Edit的Font.Size不能更新web显示时字体大小。如果特别需要，可以通过采用在Hint中设置类似{"dwstyle":"font-size:24px;"}的写法，设置字体大小等
* **注意**：设置Hint时除双引号内的部分，不要使用全角符号！，请特别注意引号和逗号
* **注意：**OnChange事件中必须有代码，可以是一个空的注释

## Edit\_\_rate（用作Rate评分控件）



* **注**：用作验证码时需要设置其HelpKeyword='rate'
* 支持OnChange事件
* 取值 ：

直接取类似 Edit1.Text

* 设置：

直接赋值类似 Edit1.Text

* 注意：Text应为0-5之间的带1位小数的字符串，如：1.6,3.5等

## Image

* 可以采用设置Hint为类似{"radius":"50%"}或{"radius":"5px"}来设置圆角
* 采用设置Hint中的href来设置新窗口跳转网址，例如：

{"href":"/hello"}，

hrefself来设置当前窗口跳转网址，例如：

{"hrefself":"/hello"}

* 采用Hint中的src来设置图片源，如：{"src":"media/images/mn/1.jpg"}
* 设置IncrementalDisplay为True时，点击图片可进行放大/旋转预览；
* 如果在程序中动态指定src，由于浏览器缓存的原因可能显示不更新。可以通过清除缓存，也可以通过类似

Hint := '{"src":"media/a.jpg?time='+IntToStr((DateTimeToUnix(Now)-86060)\*1000)+';"}'；

来强制刷新显示。

## Label

* Delphi界面效果和网页效果会有一些区别，宽度和高度最好都留一些余量
* 支持OnClick事件
* 支持OnMouseEnter事件
* 支持OnMouseLeave事件
* 如果需要强制换行，可以在Caption中增加<br>,

如：delphi是一个很棒的开发工具！我喜欢！

可以写成：delphi是一个很棒的开发工具！<br>我喜欢！

## Label\_\_cool（用作艺术字）

* **注**：用作验证码时需要设置其HelpKeyword为'cool'
* 用HelpContext区分多种效果

0：空心文字

1：空心+阴影

2：空心+叠加

3：立体文字

4：发光文字

5：彩色渐变文定

6：动态变色文字

7：阴影文字

8：霓虹灯效果

9：实心文字

## Label\_\_captcha（用作验证码）

* **注**：用作验证码时需要设置其HelpKeyword为'captcha'
* 注：把AutoSize 设置为false
* 直接读Caption得到验证码
* 点击可直接刷新，无需要代码
* 用类似以下代码主动刷新验证码(uses dwBase)

dwRunJS('document.getElementById("'+dwPrefix(Label1)+Label1.Name+'").click();',self);

## ListBox

* 支持Enabled/Visible/Left/Top/Width/Height
* 支持OnClick事件;
* 支持单选/多选;
* OnShow后动态改变Items需要执行DockSite:=True;
* Hint中设置{"dwstyle":"overflow-y:hidden"}可以去掉垂直滚动条; {"dwstyle":"overflow-y:auto"} 可以根据当前元素自动显示或不显示滚动条

## ListView\_\_mobile

ListView\_\_mobile用于

### 10.16.1 基本属性

* 设置其HelpKeyword为mobile

### 10.16.2 Hint属性

* module为整形JSON数组，共5位，分别表示增删改查移，如

”module”:[0,1,1,1,1]

* cardstyle为单个卡片的额外style，如

"cardstyle": "background:rgb(240,240,240);border:solid 1px #fdfdfd;"

* dwstyle为ListView的额外style，如

"dwstyle": "border-bottom:solid 1px #dcdfe6;"

### 10.16.3 TListColumn的Caption属性

* caption字符串型，为当前项在新增或编辑时的标题，如：

”caption”:”名称”

* set整型JSON数组，用于新增或编辑时多控件同步，如修改了产品型号，产品的名称项对应改变；
* left整型，为当前项的左边距；
* top整型，为当前项的上边距；
* width整型，为当前项的宽度；
* height整型，为当前项的高度；
* right整型，为当前项的右边距；
* bottom整型，为当前项的下边距；
* default整型，为当前整型项的默认值；
* fontname字符串型，为当前项的字体名称
* fontsize整型，为当前项的字体大小
* fontcolor字符串型，为当前项的字体颜色，如#dcdfe6
* fontbold字符串型，为当前项的字体粗体，如bold

|  |  |
| --- | --- |
| **值** | **描述** |
| normal | 默认值。定义标准的字符。 |
| bold | 定义粗体字符。 |
| bolder | 定义更粗的字符。 |
| lighter | 定义更细的字符。 |
| * 100 * 200 * 300 * 400 * 500 * 600 * 700 * 800 * 900 | 定义由粗到细的字符。400 等同于 normal，而 700 等同于 bold。 |
| inherit | 规定应该从父元素继承字体的粗细。 |

* fontdecoration字符串型，为当前项的字体装饰

|  |  |
| --- | --- |
| **值** | **描述** |
| none | 默认值。规定 text-decoration 没有线条。 |
| underline | 规定在文本下方显示线条。 |
| overline | 规定在文本上方显示线条。 |
| line-through | 规定显示横穿文本的线条。 |
| initial | 将此属性设置为其默认值。参阅 [initial](https://www.w3school.com.cn/cssref/css_initial.asp)。 |
| inherit | 从其父元素继承此属性。参阅 [inherit](https://www.w3school.com.cn/cssref/css_inherit.asp)。 |

* type字符串型，为当前项的类型

|  |  |
| --- | --- |
| **值** | **描述** |
| string | 默认值。显示为字符串型 |
| boolean | 显示为布尔型 |
| image | 显示为图片 |
| integer | 显示为数值 |
| button | 显示为按钮 |
| rate | 显示为评价 |

"

### 10.16.3 其他

## MainMenu

* 采用dwSetCompLTWH函数设置Left/Top/Width/Height （默认：0，0，600，50）
* 采用dwSetMenuDefault函数设置默认选中的菜单项，

如：dwSetMenuDefault(MainMenu,'1-0-2');

注：序号从0开始，每层之间用-隔开

* 默认Left:0;Top:0;Width:600;Height:38
* AutoHotKeys 必须设置为 maManual
* AutoMerge为是否折叠菜单
* 事件必须写在菜单项的OnClick中
* ParentBiDiMode为True时为水平显示,为True时垂直显示
* 不要使用属性Tag/DesignInfo
* 在子菜单项的Hint中设置类似{"dwstyle":"padding-left:30px;"}可以调节菜单左边距；
* 第1个菜单项的Hint用于菜单配置信息
* 其中前景色，字体色等信息；

{"background-color":"#FF0000","text-color":"#0000FF","active-text-color":"#FFFFFF"}

* 如果有dwloading项，如Hint为{"dwloading":"1"}，则点击当前MainMenu中所有菜单项时先显示”loading...“标识。关闭该标识可以在菜单项执行事件中加入以下代码即可在事件执行完成后关闭”loading...“标识。

dwRunJS('this.dwloading=false',self);

* ImageIndex见dwVars中的dwIcons常量和ElementUI中图标的定义，网址：<https://element.eleme.cn/#/zh-CN/component/icon>





## MediaPlayer

本控件有2种模式：

* 普通模式（HelpKeyword <> 'm3u8'）,支持的视频格式有：
* MP4 = MPEG 4文件使dao用 H264 视频zhuan编解码器和AAC音频shu编解码器
* WebM = WebM 文件使用 VP8 视频编解码器和 Vorbis 音频编解码器
* Ogg = Ogg 文件使用 Theora 视频编解码器和 Vorbis音频编解码器
* 视频流模式（在控件中设置HelpKeyword = 'm3u8'），支持m3u8等视频流格式，可以播放在线电视。

具体的属性有

* AutoRewind表示是否循环播放；
* Enabled表示是否自动播放（一些浏览器不支持）,**注意**不是AutoOpen；
* FileName表示当前视频位置，注意需要采用网络位置，如./ocean.mp4
* 播放，采用MediaPlayer1.EnabledButtons := [btPlay];
* 暂停，采用MediaPlayer1.EnabledButtons := [btPause];
* 设置时刻，采用类似MediaPlayer1.HelpContext := 6; //表示播放位置为6-1秒处

## MediaPlayer\_\_easyplayer（EasyPlayer）

功能说明

* 支持 MP4 播放
* 支持 m3u8/HLS 播放;
* 支持 HTTP-FLV/WS-FLV 播放;
* 支持直播和点播播放;
* 支持播放器快照截图;
* 支持点播多清晰度播放;
* 支持全屏或比例显示;
* 自带的 flash 支持极速和流畅模式;
* 自带的 flash 支持 HTTP-FLV 播放;
* 支持重连播放；

**注**：用作验证码时需要设置其HelpKeyword='easyplayer'

支持的Delphi属性有：

* AutoRewind表示是否循环播放；
* AutoOpen表示是否自动播放（一些浏览器不支持）；
* FileName表示当前视频位置，注意需要采用网络位置，如

https://ablive.easyvaas.com/live/633dc2c85fc2050afb407a9e7d22f65b.flv

支持的Hint属性有：

| **参数** | **说明** | **类型** | **默认值** |
| --- | --- | --- | --- |
| video-title | 视频右上角显示的标题 | String | - |
| snap-url | 视频封面图片 | String | - |
| live | 是否直播, 标识要不要显示进度条 | Boolean | true |
| speed | 是否显示倍速播放按钮。注意：当live为true时，此属性不生效 | Boolean | true |
| alt | 视频流地址没有指定情况下, 视频所在区域显示的文字 | String | 无信号 |
| muted | 是否静音 | Boolean | false |
| aspect | 视频显示区域的宽高比 | String | 16:9 |
| isaspect | 视频显示区域是否强制宽高比 | Boolean | true |
| loading | 指示加载状态, 支持 sync 修饰符 | String | - |
| fluent | 流畅模式 | Boolean | true |
| stretch | 是否不同分辨率强制铺满窗口 | Boolean | false |
| show-custom-button | 是否在工具栏显示自定义按钮(极速/流畅, 拉伸/标准) | Boolean | true |
| isresolution | 是否在播放 m3u8 时显示多清晰度选择 | Boolean | false |
| isresolution | 供选择的清晰度 "yh,fhd,hd,sd", yh:原始分辨率 | fhd:超清，hd:高清，sd:标清 | - |
| resolutiondefault | 默认播放的清晰度 | String | hd |

参考Hint值：

{"muted":false,"alt":"no valid singal"}

## Memo

* 支持OnChange事件
* 支持OnEnter事件
* 支持OnExit事件
* 支持OnMouseEnter事件
* 支持OnMouseLeave事件

## Memo\_\_html（用于直接显示HTML代码）

* **注**：用作验证码时需要设置其HelpKeyword='html'
* Memo的Text为拟显示的HTML代码。

**注**：请谨慎使用！ 注意HTML代码的正确性！

## Memo\_\_echarts（用于显示ECharts图表）

* **注**：用作验证码时需要设置其HelpKeyword='echarts'
* **注：**请设置ScrollBars为ssBoth，以避免超长字符串自动换行造成错误！
* Memo的Text为拟显示的echarts对应图表中代码配置的option部分。

## Memo\_\_uses（用于额外引用js/css）

* **注**：用作验证码时需要设置其HelpKeyword='uses'
* **注：**请设置ScrollBars为ssBoth，以避免超长字符串自动换行造成错误！
* Memo的Text为拟额外引用的js/css文件，参考格式如下

<script src="dist/axios.min.js" type="text/javascript"></script>

<link rel="stylesheet" type="text/css" href="dist/theme-chalk/index.css" />

<script src="dist/\_easyplayer/EasyPlayer-element.min.js"></script>

## Memo\_\_usescomp（用于额外引用控件的js/css）

## Memo\_\_markdown（用作MarkDown编辑器）

* **注**：用作MarkDown编辑器时需要设置其HelpKeyword='markdown’；

属性对应：

* ReadOnly有效；
* ShowHint表示是否显示工具栏，为True隐藏实时预览；
* ParentBiDiMode表示是否显示编辑器，为True时显示编辑器，否则为仅显示；

Hint扩展属性

* 自定义工具栏：参考

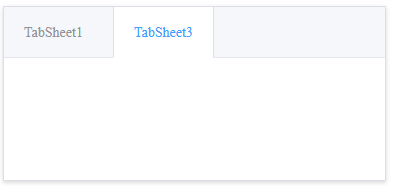
{"toolbaricons":["undo", "redo", "|", "bold", "hr", "|", "preview", "watch", "|", "fullscreen", "info", "testIcon", "testIcon2", "file", "faicon", "||", "watch", "fullscreen", "preview", "testIcon"]}

所有可选项有：

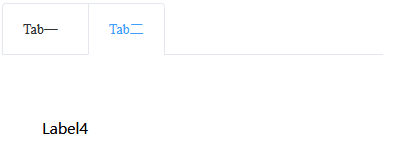
"undo","redo","|","bold","del","italic","quote","ucwords" ,"uppercase","lowercase","|","h1","h2","h3","h4","h5","h6","|","list-ul","list-ol","hr","|","link","reference-link", "image", "code","preformatted-text","code-block","table","datetime","emoji","html-entities","pagebreak","|","goto-line","watch","preview","fullscreen","clear","search","|", "help","info","releaseIcon", "index"

## PageControl

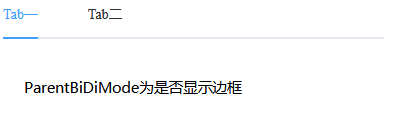
* ParentBiDiMode和ParentShowHint联合配置是否显示边框
* ParentBiDiMode=True且ParentShowHint=True时，则显示全边框，如



* ParentBiDiMode=True且ParentShowHint=False时，则显示顶部边框，如



* ParentBiDiMode=False时，不论ParentShowHint值，都不显示边框，如



* 设置Hint为{"dwattr":"editable"}，可以在TabSheet的标题处显示关闭按钮和新增按钮，当按下按钮时，激活OnEndDock事件，类似：

procedure TForm1.PageControl1EndDock(Sender, Target: TObject; X, Y: Integer);

其中参数：

* X = 1 为新增Tab ;
* X = 0 为删除Tab,待删除Tab的序号为Y（从0开始）
* 设置Hint为{"dwattr":":stretch=true"}

这样所有页面宽度会平均分布

* 设置Hint为{"dwattr":"closable"}，可以在TabSheet的标题处显示关闭按钮，当按下关闭按钮时，激活OnEndDock事件，参数和以上相同。
* 可以设置特定页面显示关闭按钮，即设置TabSheet的Hint为{"dwattr":"closable"}即可。

注意：此处不要设置PageControl的Hint为{"dwattr":"closable"}或{"dwattr":"editable"}

## PageControl\_\_timeline（用作时间线）

* **注**：PageControl用作时间线时需要设置其HelpKeyword='timeline’，如下图



* 每个TabSheet为一个时间节点
* Tag为时间值
* Caption为时间节点标题（黑体）
* 可以在Hint中通过类似{"color":"#FF0000"}来设置节点颜色
* 每个TabSheet中可以有若干Label或Memo，按创建顺序显示为标题下面的文本
* TabSheet中不能有其他控件，否则容易造成混乱！！！
* 各时间节点的顺序，根据TabSheet激活的顺序，也就是后激活的TabSheet（时间节点）排在后面。

如果需要正序，可以先执行类似代码  
    for var I := 0 to PageControl1.PageCount-1 do begin  
        PageControl1.ActivePageIndex    := I;  
    end;  
如果需要逆序，可以先执行类似代码  
    for var I := PageControl1.PageCount-1 downto 0 do begin  
        PageControl1.ActivePageIndex    := I;  
    end;

## PageControl\_\_carousel（用作轮播组件）

* **注**：PageControl用作轮播组件时需要设置其HelpKeyword='carousel’；
* 每个TabSheet为一个页面

## Panel

* BorderStyle为bsSingle，显示边框，且为圆角；否则无框直角
* Caption属性无效，实现显示时不显示Caption；
* HelpContext用来设置透明度，0：不透明，2~9：半透明，10：完全透明；
* 通过更改Hint来改变圆角半径

{"radius":"5px"}

{"radius":"5px 5px 0px 0px"}

数值有三种表示方法：px、%、em，一般推荐用px；

如果你在 border-radius 属性中只指定一个值，那么将生成 4 个 圆角。

但是，如果你要在四个角上一一指定，可以使用以下规则：

* 四个值: 第一个值为左上角，第二个值为右上角，第三个值为右下角，第四个值为左下角。
* 三个值: 第一个值为左上角, 第二个值为右上角和左下角，第三个值为右下角
* 两个值: 第一个值为左上角与右下角，第二个值为右上角与左下角
* 一个值： 四个圆角值相同
* 可以通过设置其Hint属性dwstyle完成设置Panel单独边框，类似

{"dwstyle":"border-bottom:solid 2px #aaa;border-top:solid 2px #aaa;"}

本例完成实现边框： 显示下边框，样式为solid, 宽度为2，颜色 为#aaa

{"dwstyle":"border:solid 1px #DCDFE6;"}

本例完成实现边框： 显示所有边框（上下左右），样式为solid, 宽度为2，颜色 为#aaa

{"dwstyle":"border-bottom:solid 2px #800;border-top:dashed 1px #008;"}

本例完成实现边框： 显示上边框和下边框，其中下边框样式为solid, 宽度为2，颜色 为#800，

上边框样式为dashed虚线, 宽度为1，颜色 为#008

其实，dwstyle属性还可以实现许多其他的功能，html中div 元素中的style 属性可以实现的功能都可以通过dwstyle来设置实现

* 常用的左侧有圆角，右侧没有，则可以

{"radius":"5px 0 0 5px"}

如图



* 左侧无圆角，右侧有，则可以类似

{"radius":"0 5px 5px 0"}

如图



* **注意** : Panel的Enabled不能导致子控件的Enabled
* 当Color为clNone时，显示为透明色；
* 可以采用dwattr来设置渐变色，如：

从上到下渐变

{"dwstyle":"background-image: linear-gradient(#7A88FF, #7AFFAF);"}



从左到右渐变  
{"dwstyle":"background-image: linear-gradient(to right , #7A88FF, #7AFFAF);"}



* 本地执行javascript代码（**注：本功能需要熟练掌握javascript知识，请谨慎使用！**）

可以在Hint中添加可在本地执行的javascript代码，执行时机分别为OnEnter,OnExit,OnClick

类似：

{"onenter":"Panel1\_\_col='#ccc';","onexit":"Panel1\_\_col='#eee';"}

表示当鼠标进入时，置Panel1的color为#ccc；

当鼠标退出时，置Panel1的color为#eee；

或者

{"onclick":"Panel4\_\_col='#bbb';"}

表示当鼠标单击时，置Panel1的color为#bbb；

可以控制的变量有：

Panel1\_\_top : 上边距

Panel1\_\_lef : 左边距

Panel1\_\_wid : 宽

Panel1\_\_hei : 高

Panel1\_\_vis : 可见性

Panel1\_\_col : 颜色

其中Panel1需要根据当前控件名相应更改，支持在控件A中控制另一控件B的属性。

## Panel\_\_iframe（用作iframe，可用来显示pdf）

* BorderStyle为bsSingle时，显示边框；否则无边框;
* Hint中采用src来控制iframe的src，如{"src":"/a.html"}
* 显示pdf时采用类似{"src":"/dist/pdf/web/viewer.html?file=c.pdf"}，其中c.pdf位于runtime/dist/pdf/web目录

## Panel\_\_badge（用作徽章Badge）

* **注**：需要设置其HelpKeyword为badge
* Caption为右上角显示数值，一般可以为32，99+，new等
* 如果仅显示圆点，可以在设置Hint为{"dwattr":"is-dot"}
* 可以通过设置type来设置颜色，类似Hint为{"type":"primary"},其它还有

{"type":"primary"}

{"type":"success"}

{"type":"info"}

{"type":"warning"}

{"type":"danger"}

{"type":"text"}

## Panel\_\_button（用作SpeedButton）

* **注**：需要设置其HelpKeyword为button
* 采用设置Hint中的src来设置图片源，例如：

{"src":"media/images/kehu.png"}

* 采用设置Hint中的href来设置新窗口跳转网址，例如：

{"href":"/hello"}，

hrefself来设置当前窗口跳转网址，例如：

{"hrefself":"/hello"}

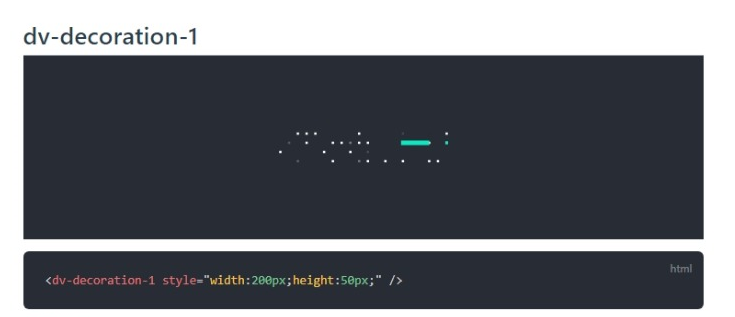
## Panel\_\_dvbox（用作大屏显示的边框）

* **注**：用作验证码时需要设置其HelpKeyword为dvbox
* **注**： 共有11种效果，通过helpcontext区分，其中默认为效果1，其余对应helpcontext数字对应的效果（其中可以4，5，8可以设置为负值，显示水平反向效果）。效果见

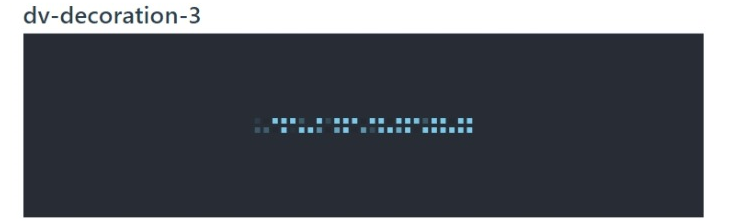
http://datav.jiaminghi.com/guide/borderBox.html#dv-border-box-1

## Panel\_\_dvdeco（用作大屏显示的装饰）

* **注**：用作验证码时需要设置其HelpKeyword为'dvdeco'
* 有两个颜色设置，分别为Color 和 Font.Color，请注意区分；
* 如果有文本显示，颜色默认采用Font.Color，也可以通过设置Hint为类似{"fontcolor":"#0f0"}修改
* 共有12种效果，通过helpcontext区分，其中默认为效果1，其余对应helpcontext数字对应的效果（其中可以2，4，8可以设置为负值，显示水平反向效果）。效果图分别如下：













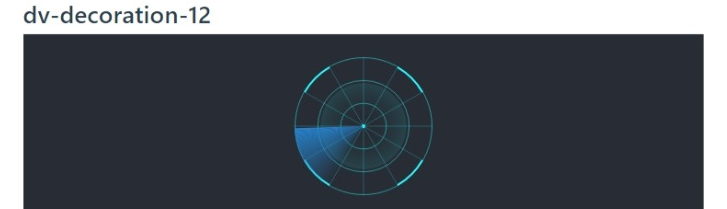












* 点击可直接刷新，无需要代码
* 用类似以下代码主动刷新验证码(uses dwBase)

dwRunJS('document.getElementById("'+dwPrefix(Label1)+Label1.Name+'").click();',self);

## ProgressBar

## RadioButton

## RadioGroup

* Color为选中颜色，建议$00FF9E40

## ScrollBox

* 其内**必须有且仅有一个Panel**，该Panel的align为alTop，其他控件请放到该Panel中；
* 滚动条滚动时自动激活OnEndDock事件，其中X为滚动值，Y为方向，1：向下，-1向上
* 滚动条滚动停止时（200毫秒内无变化）自动激活OnMouseUp事件，其中X为滚动值，
* HelpContext可以用来防止滚动条事件频繁激活，滚动值大于HelpContext时才激活OnEndDock事件。例如：HelpContext为100，也就是当滚动值（相对上次事件时的位置）大于100时才激活 OnEndDock事件
* 主要控制滚动位置，可以使用类似以下代码：

dwRunJS('this.$refs['''+dwFullName(ScrollBox1)+'''].wrap.scrollTop =200;',self);

## Shape(用于扫描二维码)

* Pen.Width为扫描框边框宽度
* Pen.Color为扫描框边框颜色
* 可以通过Hint设置视频窗口的显示情况

如{"dwstyle":"object-fit: contain;"}，其中object-fit 属性可接受如下值：

fill - 默认值。调整替换后的内容大小，以填充元素的内容框。如有必要，将拉伸或挤压物体以适应该对象。

contain - 缩放替换后的内容以保持其纵横比，同时将其放入元素的内容框。

cover - 调整替换内容的大小，以在填充元素的整个内容框时保持其长宽比。该对象将被裁剪以适应。

none - 不对替换的内容调整大小。

scale-down - 调整内容大小就像没有指定内容或包含内容一样（将导致较小的具体对象尺寸）

* 开始扫描采用函数dwSetZXing(AShape,AValue);

输入参数：

AShape类型为TShape,为当前TShape对象；

AValue类型为整型，备用

* 当扫描识别完成后自动激活TShape对象的OnEndDock事件，并通过类似的代码取得识别的值：

dwISO8859ToChinese(dwUnEscape(dwGetProp(Shape1,'scanvalue')))

在此代码后，应该加上以下代码以用于二次扫描

Shape1.HelpKeyword := ‘’;

* 可以用类似代码关闭摄像头

dwStopZXing(Shape1,0);

**注**：使用扫描识别二维码/条形码需要服务器采用https

## SpeedButton

用于生成图标居上的按钮

* 采用设置Hint中的src来设置图片源，例如：

{"src":"media/images/kehu.png"}

* 采用设置Hint中的href来设置新窗口跳转网址，例如：

{"href":"/hello"}，

hrefself来设置当前窗口跳转网址，例如：

{"hrefself":"/hello"}

* Flat为False为有外框，为True时为无外框；
* 用Layout来控制类型
* blGlyphLeft 为普通类型



* blGlyphRight 右上显示三角形，颜色为groupindex

(常用红色为255，蓝色为16711680)



* blGlyphTop 右上显示显示Badge标记，字体色为白色，数量为Spacing, 最大限制为Margin，底色通过hint中的类似Button的{"type":"primary"}等控制。



* blGlyphLeft 为右上红点提示



## SpinEdit

* Ctrl3D为False时增减按钮在左右侧；
* 如果需要显示外框可以设置Hint为

{"dwstyle":"border:solid 1px #eee;border-radius:2px"}

## StaticText

TStaticText用于链接，其Hint属性中有3项

* href用来标志链接地址，如：http://www.163.com
* type用于标志链接类型（颜色），选项有：primary,success,warning,danger,info。默认为primay
* target用来设置是否新标签页打开，"\_blank"表示在新窗口打开，"\_self"表示在当前窗口打开，如：{"href":" http://www.163.com ","target":"\_blank"}
* ParentFont=True时为系统配置的字体（字体字号颜色样式），否则为当前设置
* 设置边框，设置Hint为{"dwstyle":"border: 1px solid #DCDFE6;"}

## StringGrid

* Height应为RowCount \* DefaultRowHeight+1，以保持最佳显示效果
* BorderStyle=bsNone时为无边框
* DefaultDrawing=False时如果单元格内容超出长度，则会自动换行。

**注意**：如果内容字数太多，行高可能会被撑开

* 支持OnClick事件, 在点击时激活
* 隐藏列：可将相应列的宽度设为-1即可以隐藏，设为正值即可显示；
* 标题栏（第一行）可以通过在列标题中添加[\*left\*]或[\*center\*]或[\*right\*]设置当前列的对齐属性。列默认为左对齐（仅OnShow之前有效）。
* 标题栏可以通过设置JSON格式的字符串设置复杂列模式，如：

Cells[1,0] := '{"caption":"名称","sort":true,"filter":["净水管道","暖气片","电视机","空调","热水器","LG笔记本"]}';

该行代码表示第1列（从0开始）的标题为：名称；

显示排序按钮；

显示筛选窗口，选项有"净水管道","暖气片","电视机","空调","热水器","LG笔记本"等6个选项，如图



当设置了显示**排序**按钮时，当点击排序按钮时，StringGrid会自动激活OnGetEditMask事件。其中参数：

ACol : Integer ; 为所在列序号（从0开始）；

ARow : Integer; 为排序方向，1为升序，0为降序；

Value : string; 为标识，固定为字符串'sort'

当设置了显示**筛选**项时，当点击筛选或重置按钮时，StringGrid会自动激活OnGetEditMask事件。其中参数：

ACol : Integer ; 为所在列序号（从0开始）；

ARow : Integer; 固定为-999

Value : string; 为标识，为类似"filter":["电视机","空调"]。前面的"filter":为固定字符串，用于事件标识；方括号内的字符串列表为当前拟筛选的选项列表。当Value为"filter":{表示当前点击了”重置“按钮

* Hint中可以通过设置标题行颜色，如：

{"headerbackground":"#eee"}

* 可以通过类似设置将首列设为CheckBox

StringGrid1.Cells[0,0] := '{"caption":"日期","type":"selection"}';

设置某行的选中状态可用dwSGSetRowChecked 函数，类似

dwSGSetRowChecked(StringGrid1,2,True);

取某行的选中状态可用dwSGGetRowChecked函数，类似

dwSGGetRowChecked(StringGrid1,2)

* TStringGrid的Hint提供一种新的简洁高效(简单粗暴)的方式，即dwstyle  
  DeWeb会把dwstyle的字符直接封装到TStringGrid对应的el-table中，快速实现一些el-table已有的功能  
  如{"dwattr":"border"}表示带边框和竖线  
  {"dwattr":"stripe"}表示带斑马纹  
  {"dwattr":"show-summary"}表示带合计
* 还可以组合，比如：  
  {"dwattr":"border stripe"}表示带边框和竖线,带斑马纹
* 设置单元格样式

function dwSGSetCellStyle(

ASG : TStringGrid;

ARow,ACol : Integer;

BackGroundColor : TColor;

FontName : String;

FontSize : Integer;

FontBold : Boolean;

FontItalic : Boolean;

FontColor : TColor

):Integer;

输入值：

* + ASG : TStringGrid; 为控件名称
  + ARow,ACol : Integer;为行、列ID
  + BackGroundColor : TColor;为背景色
  + FontName : String;为字体名称
  + FontSize : Integer;为字体大小
  + FontBold : Boolean;为字体是否粗休
  + FontItalic : Boolean;为字体是否斜体
  + FontColor : TColor为字体颜色

参考：

dwSGSetCellStyle(StringGrid1,0,0,clGreen,'宋体',20,True,False,clWhite);

* 设置列样式

function dwSGSetCol(

ASG:TStringGrid;

ARow:Integer;

BackGroundColor:TColor;

FontColor:TColor

):Integer;

输入值：

* + ASG:TStringGrid;为控件名称
  + ACol:Integer;为列ID
  + BackGroundColor:TColor;为背景色
  + FontColor:TColor为字体色
* 设置行样式

function dwSGSetRow(

ASG:TStringGrid;

ARow:Integer;

BackGroundColor:TColor;

FontColor:TColor

):Integer;

输入值：

* + ASG:TStringGrid;为控件名称
  + ARow:Integer;为行ID
  + BackGroundColor:TColor;为背景色
  + FontColor:TColor为字体色

## StringGrid\_\_luck(用作大转盘抽奖)

* 需要设置HelpKeyword 为 'luck'
* RowCount为红包数量 \* DefaultRowHeight+1，以保持最佳显示效果
* 列数至少为7列

//第一列,红包标题

SG.Cells[0,I-1] := (I\*10).ToString+'元红包';

//第二列,红包区颜色

if I mod 2 = 0 then begin

SG.Cells[1,I-1] := '#f9e3bb';

end else begin

SG.Cells[1,I-1] := '#f8d384';

end;

//第三列, 红包显示文本

SG.Cells[2,I-1] := (I\*10).ToString+'元红包';

//文本顶部距离,建议10%

SG.Cells[3,I-1] := '10%';

//红包图片

SG.Cells[4,I-1] := 'media/images/luck/'+(I-1).ToString+'.png';

//红包图片宽度, 建议30%

SG.Cells[5,I-1] := '30%';

//红包图片顶部距离 , 建议45%

SG.Cells[6,I-1] := '45%';

* 抽奖完成后自动激活OnEndDock事件,其中参数X为当前奖项所在行序号(从0开始)

## StringGrid\_\_ vecandle (用作股市K线图)

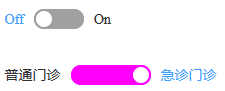
* 需要设置HelpKeyword 为 'vecandle'
* ColCount必须为6
* 第0行为参数名称，建议分别为：‘日期’，‘open’，‘close’，‘lowest’，‘highest’，‘vol’
* 从第1行开始为K线数据，分别为相应值

## Timer

* **！！！注！！！**通过设置DesignInfo来控制是否运行，其中:

1为Enabled=True；0为Enabled=False;不要通过Timer自身的Enabled来控制

## ToggleSwitch



* State为当前状态；
* ThumbColor为选中时的颜色
* SwitchWidth为选择框宽度
* StateCaptions-CaptionOff/CaptionOn为未选中/选中的标题

## TrackBar

## TrackBar\_\_page（用作Pagination分页）

* **注**：TrackBar用作轮播组件时需要设置其HelpKeyword='page’；
* Max为总数量（页数\*每页数据量）；
* Min建议为0；
* PageSize为每页显示的数据量；
* Position为当前页序号（1为首页）；
* 点击页码（包括上一页，下一页）后会触发OnChange事件
* 可以通过设计Hint的dwattr为设置显示的组件

如{"dwattr":"layout=\"prev, pager, next\""}

表示仅显示“下一页”，“页码”和“下一页”，如图



layout的选项有：sizes, prev, pager, next, jumper, ->, total, slot

* 可以通过设计Hint的dwattr为设置显示背景

{"dwattr":"background"}

如图



* 更多设置请参考

https://element.eleme.cn/#/zh-CN/component/pagination

## TrackBar\_\_wwpage（西风版Pagination分页组件）

* **注**：TrackBar用作轮播组件时需要设置其HelpKeyword='wwpage’；
* Max为总数量（页数\*每页数据量）；
* Min建议为0；
* PageSize为每页显示的数据量；
* Position为当前页序号（1为首页）；
* 点击页码（包括上一页，下一页）后会触发OnChange事件
* 在Hint中设置类似{"goto":0}可以隐藏跳转部分
* 在Hint中设置类似{"pagesize":0}可以隐藏每页条数部分
* 在Hint中设置类似{"total":0}可以隐藏总条数部分

## TreeView

* 树形控件
* 不要使用节点的StateIndex

# 系统部署

* 系统要求：需要服务器为Windows系统；
* 部署操作：把Runtime目录整体复制到服务器即可；
* 一般应在Config.json中修改端口号为80。

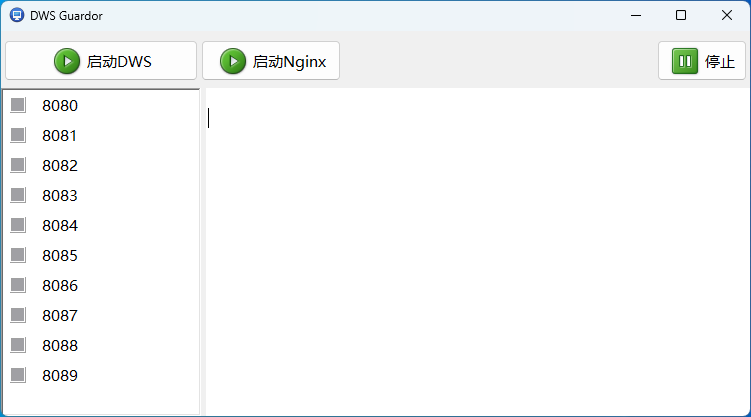
如果修改后运行时报端口被占用，可采用以下步骤解决：

1. 检查是否已有DeWebServer处于运行状态，主要是检查任务栏右下角有没有DeWebServer的图标，如果有，可以右键单击，点Exit;
2. 如果仍不能使用80端口，请使用windows命令行cmd后，输入net stop http

，以停止系统使用的80端口；

1. 如果上述不成功，可以打开Windows的IIS，停止服务

* 如果需要大并发（理论可达1000+并发），建议运行Runtime\DWSGuardor.exe



点击“启动DWS”，可以自动启动10个DWS，大概几十秒后左侧8081~8089左侧的灯变亮；



点击“启动Nginx”，可以启动nginx反向代理；

如需要关闭DWS，请点击“停止”按钮。

# 应用基本设计

* 为了尽量和网络显示一致，建议在Form默认字体为"微软雅黑，11号"，BorderStyle = bsNone；
* 大部分控件的Hint用来存放额外的信息，请勿使用；
* 不能使用ShowMessage/MessageDlg/InputQuery等所有模态窗体；

可以使用DeWeb自带的类似函数dwShowMessage, dwMessage, dwMessageDlg, dwInputQuery等；

* 所有窗体不能Show/ShowModal，

可以使用dwShowModalPro函数来弹出窗体；

# 纯数据交互（post/get）

DeWeb支持通过post/get进行纯数据交互

* 使用方法，类似：

https://delphibbs.com/dwdirect.dll?name=westwind&age=18

* 例程dwdirect位于source\demos\dwdirect

关键函数如下：

**function** dwDirectInteraction(AData:PWideChar):PWideChar;stdcall;

**function** \_Get(AParaName:String):String;

**var**

iPos : Integer; //位置,类似:name=

**begin**

Result := AData;

//删除以前的

iPos := Pos(AParaName+'=',Result);

Delete(Result,1,iPos+Length(AParaName));

//删除&后面的内容

iPos := Pos('&',Result);

if iPos>0 then begin

Result := Copy(Result,1,iPos-1);

end;

**end**;

**begin**

Result :=PWideChar(Format(' Name is %s Age is %s',[\_Get('name'),\_Get('age')]));

**end**;

* **注**：此功能支持其他语言混合编程

# 特殊函数/用法

* 打开一个新页面请使用

dwOpenUrl(self,'/discuss.dw','\_self'); //当前页面打开

dwOpenUrl(self,'/plan.dw','\_blank'); //新建页面打开

* **ShowMessage**采用dwShowMessage代替

procedure dwShowMessage(AMsg:String;AForm:TForm);

其中:

* AMsg：消息内容
* AForm：当前窗体，一般可以使用self

例：dwShowMessage('Welecome to DeWeb!',Self);

**注意**：需要提前把AMsg的字符串的HTML特殊字符替换掉，

如：“\”替换为“\\”等

如果需要自定义标题和按钮标题，则可以用 dwShowMsg函数

procedure dwShowMsg(AMsg,ACaption,AButtonCaption:String;AForm:TForm);

其中:

* AMsg：消息内容
* ACaption：消息框标题
* AButtonCaption：按钮标题
* AForm：当前窗体，一般可以使用self

例：dwShowMessage('Welecome to DeWeb!','DeWeb Demo','确定',Self);

* **dwMessage**函数，用于显示消息提示

procedure dwMessage(AMessage,AType:String;AForm:TForm);

其中:

* AMessage：消息内容
* AType：为消息类型:info/success/warning/error
* AForm：当前窗体，一般可以使用self

例：dwMessage('Welecome to DeWeb!','warning',Self);

**注意**：需要提前把AMsg的字符串的HTML特殊字符替换掉，

如：“\”替换为“\\”等

* 确定取消的**MessageDlg**采用dwMessageDlg代替

procedure dwMessageDlg(AMsg,ACaption,confirmButtonCaption,cancelButtonCaption,AMethedName:String;AForm:TForm);

其中:

* AMsg：消息内容
* ACaption：消息框标题
* confirmButtonCaption：确定按钮标题
* cancelButtonCaption：取消按钮标题
* AMethedName：返回结果后触发Form的OnStartDock中的事件标志
* AForm：当前窗体，一般可以使用self

dwMessageDlg没返回值，所以不能在当前函数/过程中处理，需要等结果返回后会自动激活Form的OnStartDock函数，

所以需要在Form的OnStartDock函数中添加事件，在事件中通过以下取得前期拟触发的事件标志

dwGetProp(Self,'interactionmethod');

通过类似以下得到返回的结果，为'1'时表示为"确定"，否则为"取消"

dwGetProp(Self,'interactionvalue');

* **InputQuery**采用dwInputQuery代替

procedure dwInputQuery(AMsg,ACaption,ADefault,confirmButtonCaption,cancelButtonCaption,AMethedName:String;AForm:TForm);

其中:

* AMsg：消息内容
* ACaption：消息框标题
* ADefault：输入框的默认值
* confirmButtonCaption：确定按钮标题
* cancelButtonCaption：取消按钮标题
* AMethedName：返回结果后触发Form的OnStartDock中的事件标志
* AForm：当前窗体，一般可以使用self

dwInputQuery没返回值，所以不能在当前函数/过程中处理，需要等结果返回后会自动激活Form的OnStartDock函数，

所以需要在Form的OnStartDock函数中添加事件，在事件中通过以下取得前期拟触发的事件标志

dwGetProp(Self,'interactionmethod');

通过类似以下得到返回的结果，为输入的内容。

dwGetProp(Self,'interactionvalue');

* **dwOpenUrl**打开网页

procedure dwOpenUrl(AForm:TForm;AUrl:String;AParams:String);

AForm:TForm 当前窗体, 一般用self

AUrl: String， 网址

AParams: String; '\_blank'是新窗口打开，'\_self'是当前窗口打开

* **dwSetCookie**设置Cookie

function dwSetCookie(AForm:TForm;AName,AValue:String;AExpireHours:Double):Integer

AForm:TForm 当前窗体, 一般用self

AName: String， Cookie名称

AValue: String; Cookie值

AExpireHours:Double 有效期（小时）

* **dwSetCookiePro**设置域名通用Cookie

function dwSetCookiePro(AForm:TForm;AName,AValue,APath,ADomain:String; AExpireHours : Integer ) : Integer;

AForm:TForm 当前窗体, 一般用self

AName: String， Cookie名称

AValue: String; Cookie值

APath:String;

ADomain:String;

AExpireHours:Double 有效期（小时）

类似：

dwSetCookiePro(self,sCookie,sValue,'/','.delphibbs.com',30\*24);

* **dwGetCookie**读取Cookie值

function dwGetCookie(AForm:TForm;AName:String):String;

AForm:TForm 当前窗体, 一般用self

AName: String， Cookie名称

* **dwRunJS** 运行javascript代码

例：alert('hello,world!');

原生的弹框函数

例：this.Button1\_\_cap='btn1';this.Label1\_\_lef='20px';

设置按钮Button1的Caption为“btn1”,设置Lable1的Left为20

* **用Javascript向Form发送消息（传递数据）**

可以在javascript中执行以下代码：

let me=this;axios.get('{"m":"interaction","i":'+this.clientid+',"t":"**varname**","v":"**varvalue**"}').then(resp =>{me.procResp(resp.data)});

以上代码中加下划线的部分可以根据需要自行设置（其他部分尽量不要修改）

varname一般用作拟传递变量的名称；

varvalue一般用作拟传递变量的值。

执行以上代码后，会自动激活Form的OnStartDock函数，

所以需要在Form的OnStartDock函数中添加事件，在事件中通过以下取得前期拟触发变量的名称

dwGetProp(Self,'interactionmethod');

通过类似以下得到拟传递变量的值。

dwGetProp(Self,'interactionvalue');

* **dwGetMobileAvailHeight**取得移动端实际可用高度函数

可以通过类似以下代码得到

Height := dwGetMobileAvailHeight(self);

* **dwSetMobileMode**设置当前屏幕显示模式为移动应用模式

function dwSetMobileMode(AForm:TForm;ADefaultWidth,ADefaultHeight:Integer):Integer;

其中:

ADefaultWidth,ADefaultHeight为电脑浏览时的默认大小。如果有一项为0，则为当前屏幕大小（只设置宽度，不设置高度）如：

//设置当前屏幕显示模式为移动应用模式.如果电脑访问，则按414x736（iPhone6/7/8 plus）显示

dwSetMobileMode(self,414,736);

* **dwUpload**文件上传函数

function dwUpload(AForm:TForm;AAccept,ADestDir:String):Integer;

参数

AForm : 窗体本身，一般用self

AAccept : 文件上传类型控制，类似"image/gif, image/jpeg"

ADestDir : 上传目录，为空时上传到服务器upload目录，有值时上传到指定目录，支持子目录，类似：'media/doc/dms/'

事件:

上传开始时，自动触发窗体的OnStartDock事件。注意：此事件激活时

dwGetProp(Self,'interactionmethod')的值为\_\_upload\_start

上传完成后，自动触发窗体的OnEndDock事件

可以通过dwGetProp(self,'\_\_upload')得到上传成功的真实文件名（如果同名文件存在，则会自动重命名），不带路径

如果在弹出式窗体中，则使用类似以下的代码

dwGetProp(TForm(self.Parent),'\_\_upload')

可以通过dwGetProp(self,'\_\_uploadsource')得到上传时原始文件名，不带路径

* **dwUploadPro**文件上传函数

function dwUpload(AForm:TForm;AAccept,AdestDir,ACapture:String):Integer;

参数:

AForm : 窗体本身，一般用self

AAccept : 文件上传类型控制，类似"image/gif, image/jpeg"

ADestDir : 上传目录，为空时上传到服务器upload目录，有值时上传到指定目录，支持子目录，类似：'media/doc/dms/'

Acapture：这个 html 的媒体属性允许开发者调用一些设备的媒体功能，主要常用值有：

* user : 前置摄像头
* camera : 相机
* environment ：后置摄像头
* 空值 ：调用麦克风和摄像

事件:

上传开始时，自动触发窗体的OnStartDock事件。注意：此事件激活时

dwGetProp(Self,'interactionmethod')的值为\_\_upload\_start

上传完成后，自动触发窗体的OnEndDock事件

可以通过dwGetProp(self,'\_\_upload')得到上传成功的真实文件名（如果同名文件存在，则会自动重命名），不带路径

如果在弹出式窗体中，则使用类似以下的代码

dwGetProp(TForm(self.Parent),'\_\_upload')

可以通过dwGetProp(self,'\_\_uploadsource')得到上传时原始文件名，不带路径

* **dwPrint**打印函数

function dwPrint(ACtrl:TControl):Integer;

参数

ACtrl为拟打印的控件，可以是DBGrid1或Button\_Edit等，全界面打印用self

* **全局刷新**

如果动态创建或删除了控件，可以采用以下代码进行全局刷新

DockSite := True;

* **指定控件强制刷新**

如果非浏览器的其他渠道更新了控件状态，可以采用以下代码对指定控件刷新

Memo1.ParentCustomHint := False; //假设该控件为Memo1

# 开发DeWeb控件

开发控件非常简单，只需要开发一个适合需要的DLL即可。

注意几点：

* 命名：工程名必须以"dw"+"控件类名"，如：dwTButton.dpr, dwTEdit.dpr等；如果该控件类型已被使用，可以通过“控件类名”+“\_\_”+“控件HelpKeyword”的方式，如dwTStringGrid\_\_ww.dpr等。注意：是2个连接下划线！
* 源码存放目录：Source\dwVcls
* 工程输出目录：Source\RunTime\Vcls\
* uses的第1个单元必须是sharemem;
* 包含以下函数，分别为：

1. **function** dwGetExtra(ACtrl:TComponent):String;stdCall;
2. **function** dwGetEvent(ACtrl:TComponent;AData:String):String;StdCall;
3. **function** dwGetHead(ACtrl:TComponent):String;StdCall;
4. **function** dwGetTail(ACtrl:TComponent):String;StdCall;
5. **function** dwGetData(ACtrl:TComponent):String;StdCall;
6. **function** dwGetAction(ACtrl: TControl): **string**; stdcall;
7. **function** dwGetMethod(ACtrl:TComponent):String;StdCall;
8. **function** dwGetMounted(ACtrl:TControl):String;stdCall;

其中：

1. dwGetExtra用于在html中增加对第三方网页控件的支持。返回值为一个JSON字符串数组格式的字符串，主要用于引入第三方的css/js，如：
2. Result  := '["<script src=\"dist/charts/echarts.min.js\"></script>",'
3. +'"<script src=\"dist/charts/lib/index.min.js\"></script>",'
4. +'"<link rel=\"stylesheet\" href=\"dist/charts/lib/style.min.css\">"]

一般建议采用JSON解析单元（uses中添加SynCommons）来处理，如下：

1. **function** dwGetExtra(ACtrl:TComponent):**string**;stdCall;
2. **var**
3. joRes     : Variant;
4. **begin**
5. //
6. **with** TStringGrid(ACtrl) **do** **begin**
7. //LuckDraw----------------------------------------------------------
8. //生成返回值数组
9. joRes    := \_Json('[]');
11. //引入对应的库
12. joRes.Add('<script src="dist/\_luckdraw/vue-luck-draw.umd.min.js"></script>');
14. //
15. Result    := joRes;
16. **end**;
17. **end**;
18. dwGetEvent用于处理控件事件

一般不需要返回值。

有2个参数

ACtrl:TComponent，控件自身

AData:String，控件信息，为JSON字符串，一般应先转换为JSON对象，比如joData。其中joData.e为事件名称，如“onchange”或“onclick”等（大小写敏感）；joData.v为值，如Edit的Value,或CheckBox的Checked等.

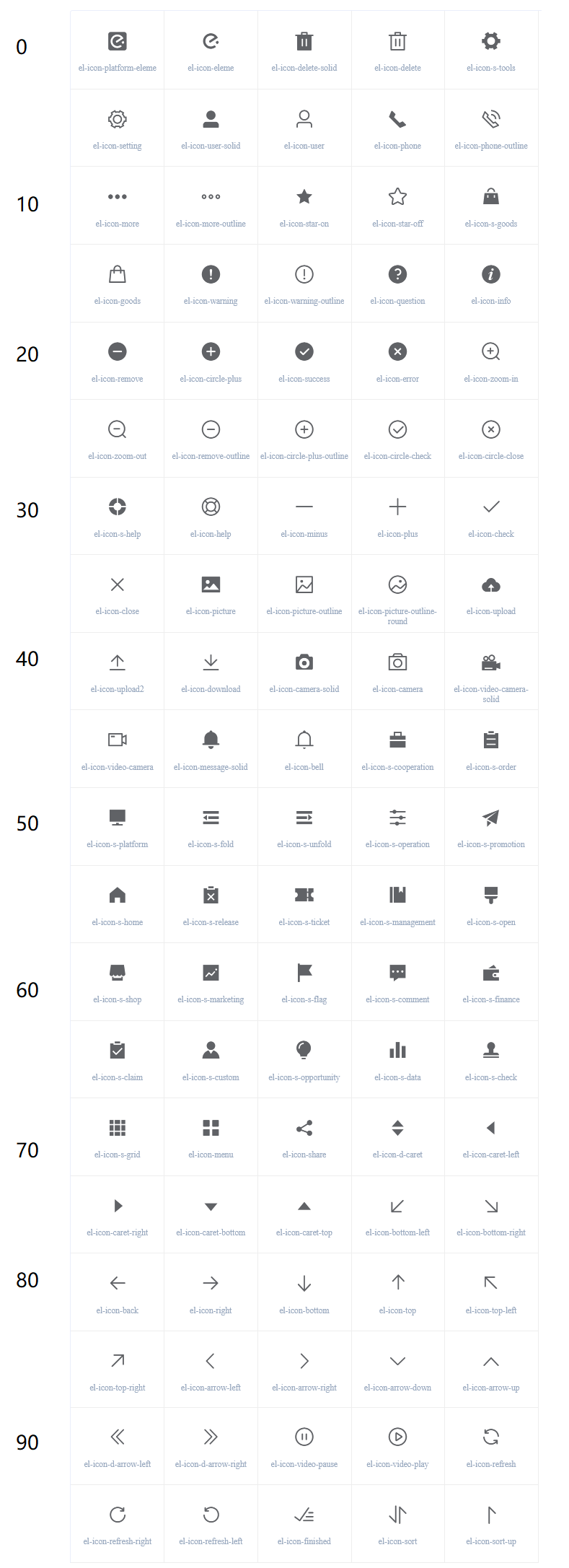
在这个函数中，需要根据这2个参数，运行相应的事件，如：

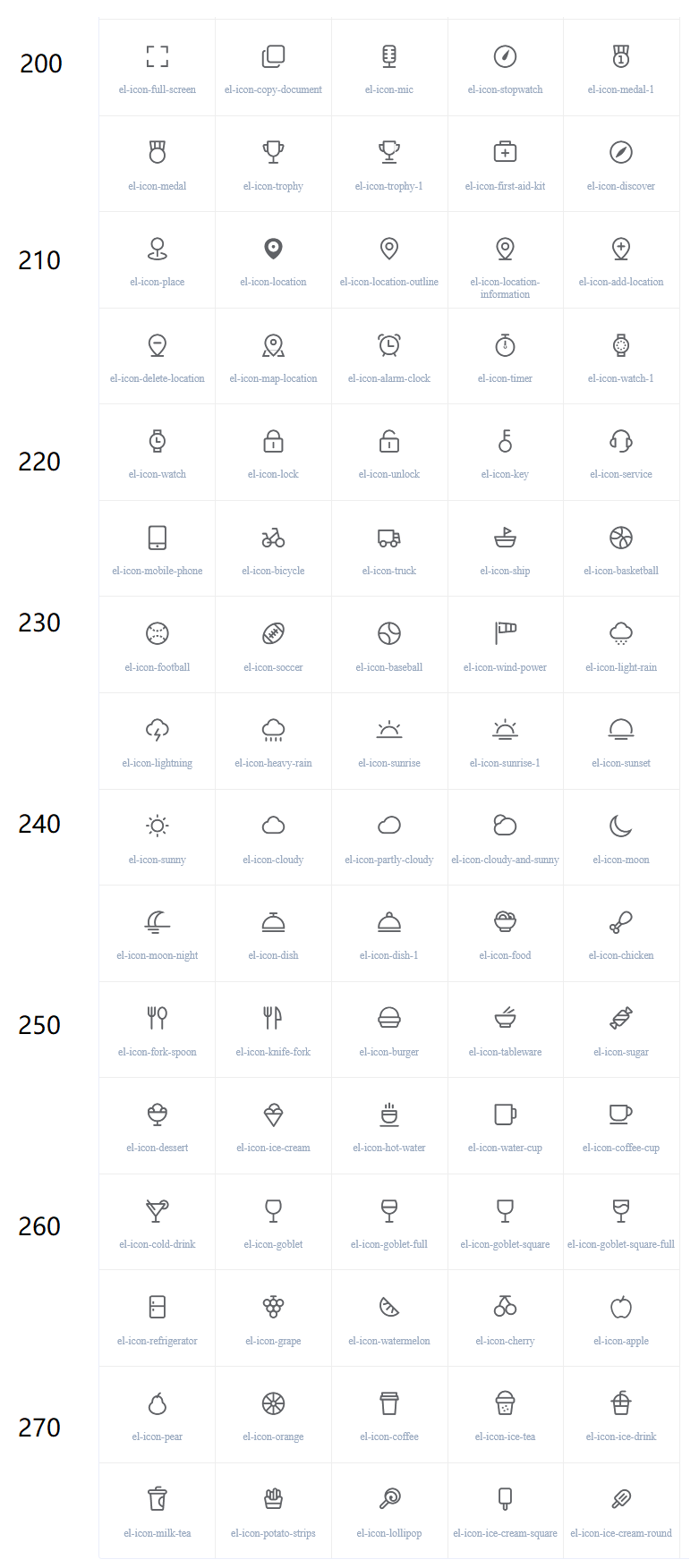
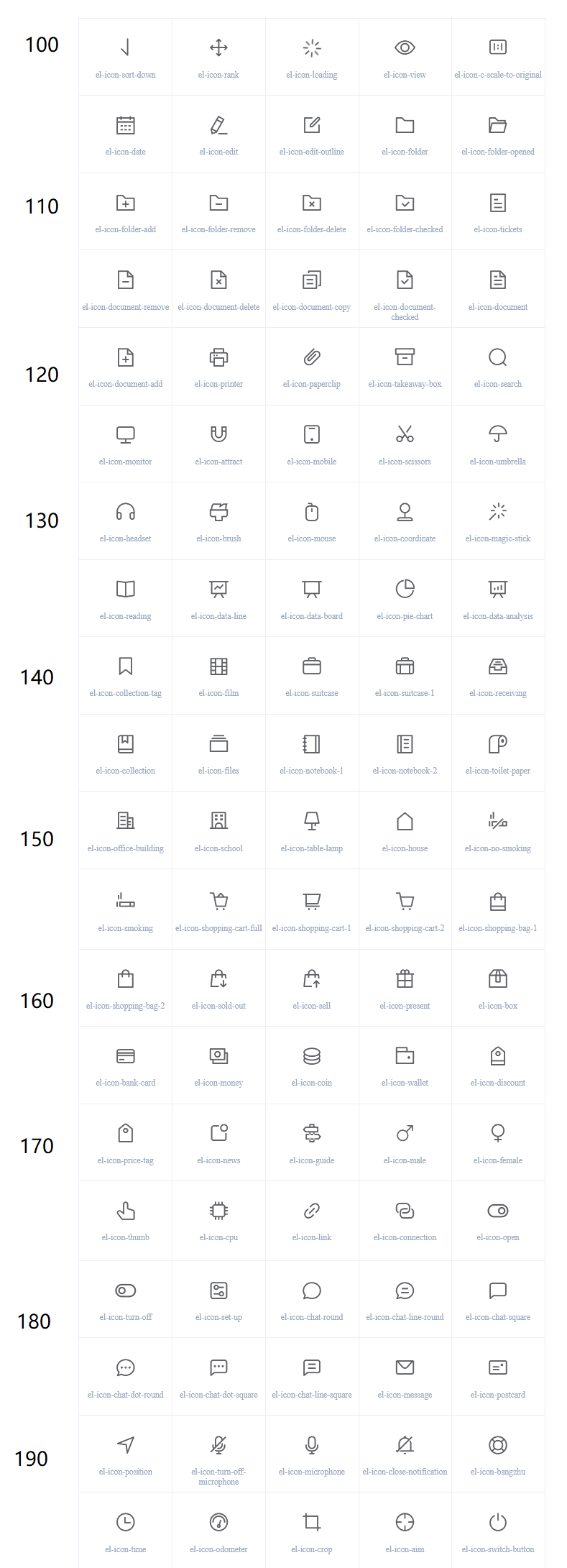
1. **function** dwGetEvent(ACtrl:TComponent;AData:String):String;StdCall;
2. **var**
3. joData  : Variant;
4. **begin**
5. //
6. joData    := \_Json(AData);
8. **if** joData.e = 'onclick' **then** **begin**
9. **if** Assigned(TButton(ACtrl).OnClick) **then** **begin**
10. TButton(ACtrl).OnClick(TButton(ACtrl));
11. **end**;
12. **end** **else** **if** joData.e = 'onenter' **then** **begin**
13. **if** Assigned(TButton(ACtrl).OnEnter) **then** **begin**
14. TButton(ACtrl).OnEnter(TButton(ACtrl));
15. **end**;
16. **end** **else** **if** joData.e = 'onexit' **then** **begin**
17. **if** Assigned(TButton(ACtrl).OnExit) **then** **begin**
18. TButton(ACtrl).OnExit(TButton(ACtrl));
19. **end**;
20. **end**;
21. **end**;
22. dwGetHead用于生成当前控件改造时的头部信息，需要根据控件对应的VUE控件来组成字符串，返回值为一个JSON字符串数组格式的字符串。参考代码如下：
23. //取得HTML头部消息
24. **function** dwGetHead(ACtrl:TComponent):**string**;StdCall;
25. **var**
26. sCode     : **string**;
27. joHint    : Variant;
28. joRes     : Variant;
29. sScroll   : **string**;
30. **begin**
31. **with** TMemo(ACtrl) **do** **begin**
33. //<处理PageControl做时间线的问题
34. **if** TMemo(ACtrl).Parent.ClassName = 'TTabSheet' **then** **begin**
35. **if** TTabSheet(TMemo(ACtrl).Parent).PageControl.HelpKeyword = 'timeline' **then** **begin**
36. joRes    := \_Json('[]');
37. //
38. Result    := joRes;
39. //
40. Exit;
41. **end**;
42. **end**;
43. //>
45. //生成返回值数组
46. joRes    := \_Json('[]');
48. //取得HINT对象JSON
49. joHint    := dwGetHintJson(TControl(ACtrl));
51. **with** TMemo(ACtrl) **do** **begin**
52. //
53. sScroll   := '';
55. sCode     := '<el-input type="textarea"'
56. +' id="'+dwFullName(Actrl)+'"'
57. +dwVisible(TControl(ACtrl))
58. +dwDisable(TControl(ACtrl))
59. +' v-model="'+dwFullName(Actrl)+'\_\_txt"'
60. +dwGetHintValue(joHint,'placeholder','placeholder','') //placeholder,提示语
61. +dwIIF(ReadOnly,' readonly','')
62. +dwGetDWAttr(joHint)
63. //style
64. +dwLTWH(TControl(ACtrl))
65. +sScroll
66. +dwGetDWStyle(joHint)
67. +'"' //style 封闭
68. //
69. +Format(\_DWEVENT,['input',Name,'escape(this.'+dwFullName(Actrl)+'\_\_txt)','onchange',TForm(Owner).Handle])
70. //+dwIIF(Assigned(OnChange),    Format(\_DWEVENT,['input',Name,'(this.'+dwFullName(Actrl)+'\_\_txt)','onchange',TForm(Owner).Handle]),'')
71. +dwIIF(Assigned(OnMouseEnter),Format(\_DWEVENT,['mouseenter.native',Name,'0','onmouseenter',TForm(Owner).Handle]),'')
72. +dwIIF(Assigned(OnMouseLeave),Format(\_DWEVENT,['mouseleave.native',Name,'0','onmouseexit',TForm(Owner).Handle]),'')
73. +dwIIF(Assigned(OnEnter),     Format(\_DWEVENT,['focus',Name,'0','onenter',TForm(Owner).Handle]),'')
74. +dwIIF(Assigned(OnExit),      Format(\_DWEVENT,['blur',Name,'0','onexit',TForm(Owner).Handle]),'')
75. +'>';
76. //添加到返回值数据
77. joRes.Add(sCode);
78. **end**;
80. //
81. Result    := (joRes);
82. **end**;
83. **end**;
84. dwGetTail用于生成当前控件改造时的尾部信息，一般与dwGetHead中的HTML要素对应即可，返回值为一个JSON字符串数组格式的字符串。
85. //取得HTML尾部消息
86. **function** dwGetTail(ACtrl:TComponent):**string**;StdCall;
87. **var**
88. joRes     : Variant;
89. **begin**
90. **with** TMemo(ACtrl) **do** **begin**
91. //<处理PageControl做时间线的问题
92. **if** TMemo(ACtrl).Parent.ClassName = 'TTabSheet' **then** **begin**
93. **if** TTabSheet(TMemo(ACtrl).Parent).PageControl.HelpKeyword = 'timeline' **then** **begin**
94. joRes    := \_Json('[]');
95. //
96. Result    := joRes;
97. //
98. Exit;
99. **end**;
100. **end**;
101. //>
103. //生成返回值数组
104. joRes    := \_Json('[]');
105. //生成返回值数组
106. joRes.Add('</el-input>');
107. //
108. Result    := (joRes);
109. **end**;
110. **end**;
111. dwGetData用于当前控件的各项关键控制参数的初始值，比如控件的Left,Top,Text等，返回值为一个JSON字符串数组格式的字符串。参考代码如下：
112. //取得Data
113. **function** dwGetData(ACtrl:TComponent):**string**;StdCall;
114. **var**
115. joRes     : Variant;
116. sCode     : String;
117. iItem     : Integer;
118. **begin**
119. **with** TMemo(ACtrl) **do** **begin**
120. //<处理PageControl做时间线的问题
121. **if** TMemo(ACtrl).Parent.ClassName = 'TTabSheet' **then** **begin**
122. **if** TTabSheet(TMemo(ACtrl).Parent).PageControl.HelpKeyword = 'timeline' **then** **begin**
123. joRes    := \_Json('[]');
124. //
125. Result    := joRes;
126. //
127. Exit;
128. **end**;
129. **end**;
130. //>
132. //生成返回值数组
133. joRes    := \_Json('[]');
134. //
135. **with** TMemo(ACtrl) **do** **begin**
136. joRes.Add(dwFullName(Actrl)+'\_\_lef:"'+IntToStr(Left)+'px",');
137. joRes.Add(dwFullName(Actrl)+'\_\_top:"'+IntToStr(Top)+'px",');
138. joRes.Add(dwFullName(Actrl)+'\_\_wid:"'+IntToStr(Width)+'px",');
139. joRes.Add(dwFullName(Actrl)+'\_\_hei:"'+IntToStr(Height)+'px",');
140. //
141. joRes.Add(dwFullName(Actrl)+'\_\_vis:'+dwIIF(Visible,'true,','false,'));
142. joRes.Add(dwFullName(Actrl)+'\_\_dis:'+dwIIF(Enabled,'false,','true,'));
143. //
144. joRes.Add(dwFullName(Actrl)+'\_\_txt:"'+dwTextToWeb(Text)+'",');
145. **end**;
146. //
147. Result    := (joRes);
148. **end**;
149. **end**;
150. dwGetAction用于当前控件的各项关键控制参数的实时动态值，返回值为一个JSON字符串数组格式的字符串。参考代码如下：
151. **function** dwGetAction(ACtrl:TComponent):**string**;StdCall;
152. **var**
153. joRes       : Variant;
154. sCode       : String;
155. iItem       : Integer;
156. joHint      : Variant;  //\_\_eventcomponent
157. sEventComp  : String;
158. **begin**
159. //得到事件源控件
160. joHint  := dwGetHintJson(TControl(ACtrl.Owner));
161. sEventComp  := '';
162. **if** joHint.Exists('\_\_eventcomponent') **then** **begin**
163. sEventComp  := LowerCase(joHint.\_\_eventcomponent);
164. **end**;
166. **with** TMemo(ACtrl) **do** **begin**
167. //<处理PageControl做时间线的问题
168. **if** TMemo(ACtrl).Parent.ClassName = 'TTabSheet' **then** **begin**
169. **if** TTabSheet(TMemo(ACtrl).Parent).PageControl.HelpKeyword = 'timeline' **then** **begin**
170. joRes    := \_Json('[]');
171. //
172. Result    := joRes;
173. //
174. Exit;
175. **end**;
176. **end**;
177. //>

180. //生成返回值数组
181. joRes    := \_Json('[]');
182. //
183. **with** TMemo(ACtrl) **do** **begin**
184. joRes.Add('this.'+dwFullName(Actrl)+'\_\_lef="'+IntToStr(Left)+'px";');
185. joRes.Add('this.'+dwFullName(Actrl)+'\_\_top="'+IntToStr(Top)+'px";');
186. joRes.Add('this.'+dwFullName(Actrl)+'\_\_wid="'+IntToStr(Width)+'px";');
187. joRes.Add('this.'+dwFullName(Actrl)+'\_\_hei="'+IntToStr(Height)+'px";');
188. //
189. joRes.Add('this.'+dwFullName(Actrl)+'\_\_vis='+dwIIF(Visible,'true;','false;'));
190. joRes.Add('this.'+dwFullName(Actrl)+'\_\_dis='+dwIIF(Enabled,'false;','true;'));
191. //
192. **if** (sEventComp <> dwFullName(Actrl)) **or** (TControl(ACtrl).ParentCustomHint=False) **then** **begin**
193. joRes.Add('this.'+dwFullName(Actrl)+'\_\_txt="'+dwTextToWeb(Text)+'";');
194. **end** **else** **begin**
195. joRes.Add('');
196. **end**
197. **end**;
198. //
199. Result    := (joRes);
200. **end**;
201. **end**;
202. dwGetMethod用于当前控件的事件Javascript代码，
203. **function** dwGetMethods(ACtrl:TControl):String;stdCall;
204. **var**
205. //
206. sCode   : **string**;
207. sFull   : **string**;
208. //
209. joHint  : Variant;
210. joRes   : Variant;
211. **begin**
212. //返回值 JSON对象，可以直接转换为字符串
213. joRes   := \_json('[]');
215. //
216. sFull   := dwFullName(Actrl);
218. //取得HINT对象JSON
219. joHint    := dwGetHintJson(TControl(ACtrl));
221. **with** TButton(ACtrl) **do** **begin**
222. //函数头部
223. sCode   := sFull+'\_\_click(event) {'#13;
224. //
225. **if** joHint.Exists('onclick') **then** **begin**
226. sCode   := sCode +joHint.onclick+#13;
227. **end**;
229. //
230. sCode   := sCode + 'this.dwevent("",'''+Name+''',''0'',''onclick'','''+IntToStr(TForm(Owner).Handle)+''');'
231. +'},';
233. //
234. joRes.Add(sCode);
235. **end**;
237. //
238. Result   := joRes;
239. **end**;
240. dwGetMounted 用于为当前控件在系统启动完成后添加代码，返回值为一个JSON字符串数组格式的字符串，参考代码如下：
241. //取得Mounted
242. **function** dwGetMounted(ACtrl:TComponent):String;StdCall;
243. **var**
244. joRes   : Variant;
245. sCode   : **string**;
246. iTab    : Integer;
247. **begin**
248. //生成返回值数组
249. joRes    := \_Json('[]');
250. //
251. joRes.Add('this.$nextTick(() => {');
252. **with** TPageControl(Actrl) **do** **begin**
253. //各页面可见性
254. **for** iTab := 0 **to** PageCount-1 **do** **begin**
255. **if** **not** Pages[iTab].TabVisible **then** **begin**
256. joRes.Add('this.$refs.'+dwFullName(Actrl)+'\_\_ref.$children[0].$refs.tabs['+IntToStr(iTab)+'].style.display = ''none'';');
257. **end**;
258. **end**;
259. **end**;
260. joRes.Add('});');
261. //
262. Result    := (joRes);
263. **end**;

# 附件1: ElementUI 图标

网址：<https://element.eleme.cn/#/zh-CN/component/icon>





# 附件2：ElementUI图标编号（ImageIndex）

1. el-icon-platform-eleme
2. el-icon-eleme
3. el-icon-delete-solid
4. el-icon-delete
5. el-icon-s-tools
6. el-icon-setting
7. el-icon-user-solid
8. el-icon-user
9. el-icon-phone
10. el-icon-phone-outline
11. el-icon-more
12. el-icon-more-outline
13. el-icon-star-on
14. el-icon-star-off
15. el-icon-s-goods
16. el-icon-goods
17. el-icon-warning
18. el-icon-warning-outline
19. el-icon-question
20. el-icon-info
21. el-icon-remove
22. el-icon-circle-plus
23. el-icon-success
24. el-icon-error
25. el-icon-zoom-in
26. el-icon-zoom-out
27. el-icon-remove-outline
28. el-icon-circle-plus-outline
29. el-icon-circle-check
30. el-icon-circle-close
31. el-icon-s-help
32. el-icon-help
33. el-icon-minus
34. el-icon-plus
35. el-icon-check
36. el-icon-close
37. el-icon-picture
38. el-icon-picture-outline
39. el-icon-picture-outline-round
40. el-icon-upload
41. el-icon-upload2
42. el-icon-download
43. el-icon-camera-solid
44. el-icon-camera
45. el-icon-video-camera-solid
46. el-icon-video-camera
47. el-icon-message-solid
48. el-icon-bell
49. el-icon-s-cooperation
50. el-icon-s-order
51. el-icon-s-platform
52. el-icon-s-fold
53. el-icon-s-unfold
54. el-icon-s-operation
55. el-icon-s-promotion
56. el-icon-s-home
57. el-icon-s-release
58. el-icon-s-ticket
59. el-icon-s-management
60. el-icon-s-open
61. el-icon-s-shop
62. el-icon-s-marketing
63. el-icon-s-flag
64. el-icon-s-comment
65. el-icon-s-finance
66. el-icon-s-claim
67. el-icon-s-custom
68. el-icon-s-opportunity
69. el-icon-s-data
70. el-icon-s-check
71. el-icon-s-grid
72. el-icon-menu
73. el-icon-share
74. el-icon-d-caret
75. el-icon-caret-left
76. el-icon-caret-right
77. el-icon-caret-bottom
78. el-icon-caret-top
79. el-icon-bottom-left
80. el-icon-bottom-right
81. el-icon-back
82. el-icon-right
83. el-icon-bottom
84. el-icon-top
85. el-icon-top-left
86. el-icon-top-right
87. el-icon-arrow-left
88. el-icon-arrow-right
89. el-icon-arrow-down
90. el-icon-arrow-up
91. el-icon-d-arrow-left
92. el-icon-d-arrow-right
93. el-icon-video-pause
94. el-icon-video-play
95. el-icon-refresh
96. el-icon-refresh-right
97. el-icon-refresh-left
98. el-icon-finished
99. el-icon-sort
100. el-icon-sort-up
101. el-icon-sort-down
102. el-icon-rank
103. el-icon-loading
104. el-icon-view
105. el-icon-c-scale-to-original
106. el-icon-date
107. el-icon-edit
108. el-icon-edit-outline
109. el-icon-folder
110. el-icon-folder-opened
111. el-icon-folder-add
112. el-icon-folder-remove
113. el-icon-folder-delete
114. el-icon-folder-checked
115. el-icon-tickets
116. el-icon-document-remove
117. el-icon-document-delete
118. el-icon-document-copy
119. el-icon-document-checked
120. el-icon-document
121. el-icon-document-add
122. el-icon-printer
123. el-icon-paperclip
124. el-icon-takeaway-box
125. el-icon-search
126. el-icon-monitor
127. el-icon-attract
128. el-icon-mobile
129. el-icon-scissors
130. el-icon-umbrella
131. el-icon-headset
132. el-icon-brush
133. el-icon-mouse
134. el-icon-coordinate
135. el-icon-magic-stick
136. el-icon-reading
137. el-icon-data-line
138. el-icon-data-board
139. el-icon-pie-chart
140. el-icon-data-analysis
141. el-icon-collection-tag
142. el-icon-film
143. el-icon-suitcase
144. el-icon-suitcase-1
145. el-icon-receiving
146. el-icon-collection
147. el-icon-files
148. el-icon-notebook-1
149. el-icon-notebook-2
150. el-icon-toilet-paper
151. el-icon-office-building
152. el-icon-school
153. el-icon-table-lamp
154. el-icon-house
155. el-icon-no-smoking
156. el-icon-smoking
157. el-icon-shopping-cart-full
158. el-icon-shopping-cart-1
159. el-icon-shopping-cart-2
160. el-icon-shopping-bag-1
161. el-icon-shopping-bag-2
162. el-icon-sold-out
163. el-icon-sell
164. el-icon-present
165. el-icon-box
166. el-icon-bank-card
167. el-icon-money
168. el-icon-coin
169. el-icon-wallet
170. el-icon-discount
171. el-icon-price-tag
172. el-icon-news
173. el-icon-guide
174. el-icon-male
175. el-icon-female
176. el-icon-thumb
177. el-icon-cpu
178. el-icon-link
179. el-icon-connection
180. el-icon-open
181. el-icon-turn-off
182. el-icon-set-up
183. el-icon-chat-round
184. el-icon-chat-line-round
185. el-icon-chat-square
186. el-icon-chat-dot-round
187. el-icon-chat-dot-square
188. el-icon-chat-line-square
189. el-icon-message
190. el-icon-postcard
191. el-icon-position
192. el-icon-turn-off-microphone
193. el-icon-microphone
194. el-icon-close-notification
195. el-icon-bangzhu
196. el-icon-time
197. el-icon-odometer
198. el-icon-crop
199. el-icon-aim
200. el-icon-switch-button
201. el-icon-full-screen
202. el-icon-copy-document
203. el-icon-mic
204. el-icon-stopwatch
205. el-icon-medal-1
206. el-icon-medal
207. el-icon-trophy
208. el-icon-trophy-1
209. el-icon-first-aid-kit
210. el-icon-discover
211. el-icon-place
212. el-icon-location
213. el-icon-location-outline
214. el-icon-location-information
215. el-icon-add-location
216. el-icon-delete-location
217. el-icon-map-location
218. el-icon-alarm-clock
219. el-icon-timer
220. el-icon-watch-1
221. el-icon-watch
222. el-icon-lock
223. el-icon-unlock
224. el-icon-key
225. el-icon-service
226. el-icon-mobile-phone
227. el-icon-bicycle
228. el-icon-truck
229. el-icon-ship
230. el-icon-basketball
231. el-icon-football
232. el-icon-soccer
233. el-icon-baseball
234. el-icon-wind-power
235. el-icon-light-rain
236. el-icon-lightning
237. el-icon-heavy-rain
238. el-icon-sunrise
239. el-icon-sunrise-1
240. el-icon-sunset
241. el-icon-sunny
242. el-icon-cloudy
243. el-icon-partly-cloudy
244. el-icon-cloudy-and-sunny
245. el-icon-moon
246. el-icon-moon-night
247. el-icon-dish
248. el-icon-dish-1
249. el-icon-food
250. el-icon-chicken
251. el-icon-fork-spoon
252. el-icon-knife-fork
253. el-icon-burger
254. el-icon-tableware
255. el-icon-sugar
256. el-icon-dessert
257. el-icon-ice-cream
258. el-icon-hot-water
259. el-icon-water-cup
260. el-icon-coffee-cup
261. el-icon-cold-drink
262. el-icon-goblet
263. el-icon-goblet-full
264. el-icon-goblet-square
265. el-icon-goblet-square-full
266. el-icon-refrigerator
267. el-icon-grape
268. el-icon-watermelon
269. el-icon-cherry
270. el-icon-apple
271. el-icon-pear
272. el-icon-orange
273. el-icon-coffee
274. el-icon-ice-tea
275. el-icon-ice-drink
276. el-icon-milk-tea
277. el-icon-potato-strips
278. el-icon-lollipop
279. el-icon-ice-cream-square
280. el-icon-ice-cream-round