Yigo语言参考

Yigo语言参考

目录

第 1 部分 公共函数参数

目录

第 1 章 公共函数

目录

1.1:ToUpper

原型：String ToUpper(value)

功能：字符串转大写。

适用平台：All

参数列表

- value 源字符串

返回值：value的大写形式。

示例：

```
// 返回值为"THIS IS A STRING"
ToUpper("This is a string")
```

1.2:ToLower

原型：String ToLower(value)

功能：字符串转小写。

适用平台：All

参数列表

- value 源字符串

返回值：value的小写形式

示例：

```
// 返回值为"this is a string"
ToLower("This is a string")
```

1.3:Trim

原型：String Trim(value)

功能：去除字符串首先的空白符，包括空格、退格、回车符。

适用平台：All

参数列表

- value 源字符串

返回值：value去除前后空白后字符串。

示例：

```
// 返回值为"a string"
Trim("   a string   ")
```

1.4:ToLong

原型：Long ToLong(value)

功能：将参数定义的数据转长整型。在value为Integer时返回其Long型值；value为Long型值时直接返回源值；value为String时，并且value为可以转为数值类型的字符串时返回其整数部分，如果value不能转化为数值，抛出格式错误异常；value为Boolean时，如果value为True返回1，否则返回0；value为BigDecimal时返回其整数部分；value为null时，返回0。

适用平台：All

参数列表

- value 源值，类型为Integer、Long、String、Boolean、BigDeciamal。

返回值：Long型值。

示例：

```
// 返回值为1
ToLong(1)

// 返回值为2
ToLong("2.22")

// 抛出异常
ToLong("a2.22")

// 返回1
ToLong(True)
```

```
// 返回2
ToLong(2.22)
```

1.5:ToInt

原型：Integer ToInt(value)

将参数定义的数据转整型。在value为Integer时返回其Long型值；value为Long时返回其Integer值；value为String时，并且value为可以转为数值类型的字符串时返回其整数部分，如果value不能转换为数值，抛出格式错误异常；value为Boolean时，如果value为True返回1，否则返回0；value不BigDecimal时返回其整数部分；value为null时，返回0。

适用平台：All

参数列表：

- value 源值，类型为Integer、Long、String、Boolean、BigDecimal。

返回值：Integer型值。

示例：

```
// 返回值为5
ToInt(5)

// 返回值为6
ToInt("6.22")

// 抛出异常
ToInt("a2.22")

// 返回0
ToInt(False)

// 返回10
ToInt(10.22)
```

1.6:ToBool

原型：Boolean ToBool(value)

将参数定义的数据转布尔类型。在value为数值类型时非0为真；在value为Boolean类型时取原值；在value为字符串时，值为"1"和"True"时为真，值为"0"和"False"时为假，其它非空为真；在value取其它类型时非null为真。

适用平台：All

参数列表：

- value 源值，类型为Integer、Long、String、Boolean、BigDecimal。

返回值：value的布尔类型值。

示例：

```
// 返回True
ToBool(True);

// 返回True
ToBool("True");

// 返回True
ToBool("adfadaf");

// 返回True
ToBool(10);
```

```
// 返回False
ToBool(0);
```

1.7:InList

原型：Boolean InList(value, c1, c2, ..cn.)

判断value的取值是否在c1,c2...cn定义的取值范围内，其中c表示的为不定长参数。

适用平台：All

参数列表：

- value 为需要判断的值；
- cn 为取值范围，不定长参数列表；

注意

这里value的类型必须和cn的类型一致，并且必须是基本类型，不能是对象类型。

返回值：Boolean类型。

示例：

```
// 返回True
InList(1, 1, 2, 3)

// 返回False
InList(1, 2, 3, 4)

// 返回True
InList('a', 'a', 'b', 'c')
```

1.8:Left

原型：String Left(s, length)

返回s最左边length个字符组成的子串。

适用平台：All

参数列表：

- s 字符串；
- length 长度；

返回值：String值。

示例：

```
// 返回"abc"
Left("abcdef", 3)
```

1.9:Right

原型：String Right(s, length)

返回s最右边的length个字符组成的子串；

适用平台：All

参数列表：

- s 字符串
- length 长度

返回值：String值。

示例：

```
// 返回"def"
Right("abcdef", 3)
```

1.10:Mid

原型：String Mid(s, pos, length)

返回s从pos位置开始的length字符的子串。

适用平台：All

参数列表：

- s 字符串
- pos 位置
- length 长度

返回值：String值。

示例：

```
// 返回"de"
Mid("abcdefg", 3, 2)
```

1.11:Length

原型：Integer Length(s)

返回s字符串的长度；

适用平台：All

参数列表：

- s 字符串

返回值：s的长度，如果s为null，返回0。

示例：

```
// 返回值为6
Length("string")
```

1.12:IndexOf

原型：Integer IndexOf(s, sub)

返回sub字符串在s中的位置。

适用平台：All

参数列表：

- s 源字符串
- sub 子串

返回值：sub在s中的位置，如果s中不存在sub子串，返回-1。

示例：

```
// 返回值为5
IndexOf("This is a string", "is")

// 返回值为-1
IndexOf("This is a string', "dog")
```

1.13:ToDecimal

原型：BigDecimal ToDecimal(value)

将value转为BigDecimal类型的值。value为Intger和Long型，返回value的BigDecimal表示值；value为String时，如果value可以转化为数值，返回该数值，否则抛出格式错误异常；value为BigDecimal时返回原值；value为Boolean时，如果value为True，返回1，否则返回0；value为null时返回0。

适用平台：All

参数列表：

- value 源值，类型为Integer、Long、String、BigDecimal、Boolean。

返回值：BigDecimal值。

示例：

```
// 返回值为1
ToDecimal(1)

// 返回值为1.11
ToDecimal("1.11")

// 抛出异常
ToDecimal("a1.11")

// 返回1.11
ToDecimal(1.11)

// 返回1
ToDecimal(True)
```

1.14:ToString

原型：String ToString(value)

将value转化为字符串形式。value可以是任何支持转字符串的数据类型，如下：

- value为Integer和Long类型时，转化为字符串表示的形式，比如123转成"123"；
- value为Boolean类型时，如果value为True，转化为"True"，否则转化为"False"；
- value为String类型时，返回原值；
- value为BigDecimal类型时，返回其字符串形式，规则参照Java语言中BigDecimal的toString规则；
- value为DataTime时，返回其字符串表示形式，格式化字符串"yyyy-MM-dd"。

- value为null时，返回""；
- value为其它类型时，返回其Java语言的toString的返回值。

适用平台：All

参数列表：

- value需要转化的值，取值为任何类型；

返回值：String类型,value的字符串表示。

示例：

```
// 返回值为"123"
ToString(123)

// 返回值为"123.33"
ToString("123.33")
```

1.15:IIF

原型：Object IIF(c, v1, v2)

根据条件返回值；

适用平台：All

参数列表：

- c 条件
- v1 值1
- v2 值2

返回值：如果c为True，那么返回v1，如果c为False，那么返回v2，如果v2未定义，返回null；

示列：

```
// 返回值为2
IIF(1==2, 1, 2)

// 返回值为null
IIF(1==2, 1)

// 返回值为"Found"
IIF(IndexOf("This is a dog", "dog")>=0, "Found", "Not Found")
```

1.16:IIFS

原型：Object IIFS(c1, v1, c2, v2, ...cn, vn ...)

返回条件-值序列中的值，从cn和vn的配对中，从左到右计算，直到cn满足条件，则返回vn的值，否则返回null。

适用平台：All

参数列表：

- cn条件
- vn值

注意

cn和vn必须配对出现，即IIFS的参数数量必须为偶数。

返回值：第一个满足条件的cn的对应的cn的值，如果没有满足的条件，返回null。

示例：

```
// 返回值为3
IIFS(1==2, 1, 1==3, 2, 1==1, 3)
```

1.17:ToLower

原型：String ToLower(s)

字符串转小写。

适应平台：All

参数列表：

- s 字符串

返回值：s的小写形式的字符串。

示例：

```
// 返回值为"this"
ToLower("This")

// 返回值为"this"
ToLower(Trim("  This "))
```

1.18:ToUpper

原型：String ToUpper(s)

字符串转大写。

适用平台：All

参数列表：

- s 字符串

返回值：s的大写形式的字符串。

示例：

```
// 返回值为"THIS"
ToUpper("This")

// 返回值为"THIS"
ToUpper(Trim("  This "))
```

1.19:IsNull

原型：IsNull(v)

判断一个值是否为null。

适用平台：All

参数列表：

- v 值

返回值：如果v为null返回True，否则返回False。

示例：

```
// 判断Mtl组件的值是否为null
IsNull(Mtl)
```

1.20:Format

原型：String Format(value, format)

根据format格式化字符串。

适用平台：All

参数列表：

- value 值，可以是整型、长整型、数值类型、布尔类型、日期类型
- format 格式，可选参数，根据value类型的不同，定义如下：
  - 数值类型，BigDecimal时为format的值同DecimalFormat格式定义
  - 整型，format无效
  - 长整型，format无效
  - 布尔类型，format无效，为真时返回true，为假时返回false
  - 日期类型，format的值同SimpleDateFormat中的格式定义

返回值：如果value为null时返回空，其中值时返回根据format转换后的字符串。

示例：

```
// 输出为仅有日期的字符串
Format(ServerDBDate(), 'yyyy-MM-dd')
```

1.21:DateAdd

原型：Date DateAdd(date, internal, number)

在日期上增加指定的时间间隔形成新的时间。

适用平台：All

参数列表：

- date 起始时间；
- interval 间隔单位，用于确定number的单位，取值如下：
  - d或dd 天
  - s 秒
  - n 分

- h 时
- m或mm 月
- yyyy 年
- q 季度
- ww 周

- number 增加的时间单位数

返回值：date增加number指定的单位时间间隔后的新时间。

示例：

```
var now = ServerDBDate();
// 取下一天
var tomorrow = DateAdd(now, 'd', 1);
```

1.22:DaysOfMonth

原型：Date DaysOfMonth(date)

取得某个日期所在月份的总天数。

适用平台：All

参数列表：

- date 日期；可选，如果不填，取当前时间。

返回值：date所在月份的总天数。

示例：

```
// 取当月总天数
var days = DaysOfMonth(ServerDBDate());
```

1.23:FirstDayOfMonth

原型：Integer FirstDayOfMonth(date)

取得某个日期所在月份的第一天。

适用平台：All

参数列表：

- date 日期；可选，如果不填，取当前时间。

返回值：date所在月份的第一天。

示例：

```
// 取得当前月份的第一天
FirstDayOfMonth(ServerDBDate())
```

1.24:LastDayOfMonth

原型：Date LastDayOfMonth(date)

取得某个日期所在月份的最后一天。

适用平台：All

参数列表：

- date 日期；可选，如果不填，取当前时间。

返回值：date所在月份的最后一天。

示例：

```
// 取得当前月份的最后一天
LastDayOfMonth(ServerDBDate())
```

1.25:DateDiff

原型：int[long] DateDiff(startDate, endDate, unit)

求两个日期之间的时间差。

适用平台：All

参数列表：

- startDate 起始时间；
- endDate 结束时间；
- unit 时间单位。取值如下：
  - yyyy 年差
  - m 月差
  - d 日差
  - h 时差
  - n 分钟差
  - s 秒差

返回值：以unit表示的时间度量为单位的时间间隔。

示例：

```
var now = ServerDBDate();
var tomorrow = DateAdd(now, 'd', 1);
// 日差
OUT(DateDiff(now, tomorrow, 'd'));
// 时差
OUT(DateDiff(now, tomorrow, 'h'));
// 秒差
OUT(DateDiff(now, tomorrow, 's'));
```

1.26:DayOfWeek

原型：Integer DayOfWeek(date)

取得某天在一周中是星期几。

适用平台：All

参数：

• date 日期；可选，如果不填，取当前时间

返回值：以数值表示的星期几，星期日为1，其它依次类推。

示例：

```
// 取得今天是星期几
var now = ServerDBDate();
OUT(DayOfWeek(now));
```

1.27:ToDate

原型：Date ToDate(s, format)

通过字符串解析日期。

适用平台：All

参数：

• s 字符串；

• format s的格式，yyyy表示年份，MM表月份，dd表示天，HH表示小时，mm表示分，ss表示秒。

返回值：如果s满足format表示的格式则返回日期。

示例：

```
// 取2015-09-03的下一天
var base = ToDate('2015-09-03', 'yyyy-MM-dd');
var tomorrow = DateAdd(base, 'd', 1);
var s = Format(tomorrow, 'yyyy-MM-dd');
OUT(s);
```

1.28:ReplaceByPos

原型：String ReplaceByPos(format, args)

根据格式字符串中占位符的位置替换内容生成新文本。

适用平台：All

参数：

• format 格式字符串，其中的占位符以{n}表示，其中n为整数，从1开始。

• args 不定长参数，用于替换相应的占位位置的占位符。

返回值：format中的占位符被替换成相应值后的字符串，比如格式为"{1}的首都是{2}"，如果替换的参数分别是"中国"和"北京"，返回的结果为"中国的首都是北京"。

示例：

```
var s = ReplaceByPos("{1}的首都是{2}", "中国", "北京");
```

第 2 部分 客户端函数参考

目录

第 2 章 表单基础函数

目录

表单处理函数为客户端提供表单基础操作功能。

2.1:LoadData

原型：Boolean LoadData()

在表单上从中间层载入当前表单的数据。此函数通过表单的配置的数据源从中间层载入数据，如果表单没有定义数据源则不作处理。

适用平台：All

参数：无

返回值：在未抛出异常的情况下，返回True，否则返回值无意义。

示例：

```
// 载入数据
LoadData()
```

2.2:SaveData

原型：Boolean SaveData()

保存表单的数据至中间层数据源。保存成功后重新加载表单的数据并将表单设置为普通状态。表单无数据源的情况下不处理。

适用平台：All

参数：无

返回值：在未抛出异常的情况下，返回True，否则返回值无意义。

示例：

```
// 保存数据
SavaData()
```

2.3:DeleteData

原型：Boolean DeleteData()

删除表单的数据，删除成功后将表单设置为普通状态。

适用平台：All

参数：无

返回值：在未抛出异常的情况下，返回True，否则返回值无意义。

示例：

```
// 删除数据
DeleteData()
```

2.4:New

原型：Boolean New(formKey, target)

新建一个指定的实体表单，formKey为指定表单的标识，如果指定的表单不是实体表单时，结果未知。新建表单后需要计算其所有组件和单元格的值，并计算可见性、可用性和检查规则。新生成的表单在界面上有三种显示方式，分别是：覆盖自身、在新的tab页中打开和显示模态窗口。

适用平台：All

参数：

- formKey 表单的标识，需要为实体类型的表单，否则新建的表单无意义；如果表单不存在，抛出指定的表单不存在异常。

- target　　打开的目标，取值为"self"(覆盖自身)、"newtab"(新tab页中打开)、"modal"(显示为模态)。可选参数，默认值为"newtab"。如果取值为self时，将当前界面置为新增状态，并不创建新的表单，此时formKey的取值无效。

返回值：在未抛出异常的情况下，返回值为True，否则返回值无意义。

示例：

```
// 新建一个表单，并在新tab页中显示
New("StockIn")

// 新建一个表单，并显示为模态窗口
New("StockIn", "modal")

// 将当前界面置为新增状态
New("", "self")
```

2.5:CopyNew

原型：void CopyNew()

从当前表单的数据中复制新增生成一个新的表单数据，清除所有系统字段的值，将表单置为新增状态。

适用平台：All

参数：无

示例：

```
// 复制新增
CopyNew()
```

2.6:Open

原型：Boolean Open(formKey, OID, target)

指定实体表单，通过对象标识打开一个表单并显示其数据。如果指定的表单不是实体表单时，结果未知。打开表单后，需要计算其所有未绑定数据源的组件和单元格的值，并计算可见性、可用性和检查规则。打开的表单在界面上有三种显示方式：覆盖自身、在新的tab页中打开和显示为模态窗口。

适用平台：All

参数：

- formKey 表单标识，覆盖为实体类型的表单，如果表单不存在，抛出指定的表单不存在异常。
- OID 对象的标识。
- target　　打开的目标，取值为"self"(覆盖自身)、"newtab"(新tab页中打开)、"modal"(显示为模态)。可选参数，默认值为"newtab"。

返回值：在未抛出异常的情况下，返回值为True，否则返回值无意义。

示例：

```
// 打开StockIn的对象标识为10001的表单
Open("StockIn", 10001)

// 打开StockIn的对象标识为10001的表单
var OID = 10001;
Open("StockIn", OID);
```

2.7:Edit

原型：Boolean Edit()

将当前的表单转为编辑状态，计算所有没有数据绑定的组件和单元格的值，计算可见性、可用性和检查规则。将表单设置为编辑状态。

适用平台：All

参数：无

返回值：在未抛出异常的情况下，返回值为True，否则返回值无意义。

示例：

```
// 将表单转为编辑状态
Edit()
```

2.8:Cancel

原型：Boolean Cancel()

取消表单的新增和编辑状态。将表单转入默认状态。

适用平台：All

参数：无

返回值：在未抛出异常的情况下，返回值为True，否则返回值无意义。

示例：

```
// 取消表单的新增和编辑状态
Cancel()
```

2.9:ReadOnly

原型：Boolean ReadOnly()

返回表单是否处于只读状态。

适用平台：All

参数：无

返回值：在表单的不是新增和编辑状态时返回True，否则返回False。

示例：

```
// 返回表单的只读状态
ReadOnly()
```

2.10:SetPara

原型：Boolean SetPara(key, value)

为表单设置指定标识的参数值。

适用平台：All

参数：

- key 参数的标识
- value 参数的值

返回值：始终返回True。

示例：

```
// 设置参数ItemKey的值为"Mtl"
SetPara("ItemKey", "Mtl")

// 设置参数OID的值为10001
SetPara("OID", 10001)
```

2.11:GetPara、Para

原型：Object GetPara(key)

返回表单的参数的值。

适用平台：All

参数：

- key 参数的标识

返回值：如果key指定的参数存在则返回参数的值，否则返回null。

示例：

```
SetPara("ItemKey", "Mtl");
// 返回值为"Mtl"
GetPara("ItemKey");

SetPara("OID", 10001);
// 返回值为10001
Para("OID");
```

2.12:GetFormCaption

原型：String GetFormCaption()

返回表单的名称。

适用平台：All

参数：无

返回值：表单的名称。

示例：

```
// 返回表单的名称
GetFormCaption()
```

2.13:GetFormAbbrCaption

原型：String GetFormAbbrCaption()

返回表单的缩写名称。

适用平台：All

参数：无

返回值：返回表单的缩写名称。

示例：

```
// 返回表单的缩写名称
GetFormAbbrCaption()
```

2.14:GetStatusItems

原型：Object GetStatusItems()

取得表单的状态集合，返回值为以PairItemList表示的状态项集合。表单的状态集合按照的优先级取得：表单自身定义的状态集合、工程的状态集合、应用的状态集合。

适用平台：All

参数：无

返回值：状态项集合。

示例：

```
// 返回表单的状态项集合，通常使用在下拉框的项目列表函数中
GetStatusItems()
```

2.15:StatusValue

原型：Integer StatusValue(statusKey)

给定状态的标识返回状态的值。

适用平台：All

返回值：statusKey对应的状态值，如果statusKey不存在，返回null。

示例：

```
// 有以下状态定义：
// Init 0
// Processing 1
// Audited 2
// 返回值为1
StatusValue("Processing")
```

2.16:SetResult

原型：Boolean SetResult(value)

设置表单的结果值；

适用平台：All

参数：

- value 值

返回值：始终返回True，无意义。

示例：

```
var successed = False;
// 其它处理...
if ( successed ) {
    // 设置返回值为True
    SetResult(True);
} else {
    // 设置返回值为False
    SetResult(False);
}
```

2.17:GetResult

原型：Object GetResult()

取得表单的结果值，该结果值为SetResult函数设置。

适用平台：All

参数：无

返回值：表单的结果值。

示例：

```
SetResult(1);
// 其它处理
// 取得之前的结果
if ( GetResult() == 1 ) {
    // 处理1
} else {
    // 处理2
}
```

2.18:Close

原型：Boolean Close()

关闭表单界面。这个函数首先确认是否需要关闭，如果确认关闭则关闭，否则什么都不做。

适用平台：All

参数：无

返回值：无异常的情况下，始终为True，否则无意义。

示例：

```
// 关闭表单
Close()
```

2.19:Show

原型：Boolean Show(key, target)

创建一个表单并显示。创建的表单根据target确定以何种方式显示。新建的表单在界面上有三种显示方式：覆盖自身、在新的tab页中打开和显示为模态窗口。

适用平台：All

参数：

- key 表单的标识

- target 打开的目标，取值为"newtab"(新tab页中打开)、"modal"(显示为模态)。可选参数，默认值为"newtab"。

返回值：在无异常的情况下始终为True，否则返回值无意义。

示例：

```
// 以新tab页的方式显示一个表单
Show("StockIn")

// 以模态方式显示一个表单
Show("StockIn", "modal")
```

2.20:GetCaption

原型：String GetCaption(key)

取得组件的显示值。

适用平台：All

参数：

- key 组件的标识

返回值：组件的显示值。

示例：

```
// 数值编辑框，千分位，值为1234.22
// 返回值为1,234.22
GetCaption("Num")
```

2.21:ShowModal

原型：Boolean ShowModal(key)

以模态方式显示一个表单，同Show功能相似，只是确定以模态显示。

适用平台：All

参数：

- key 表单的标识。

返回值：在无异常的情况下始终为True，否则返回值无意义。

示例：

```
// 模态显示查询界面CondStockIn
ShowModal("CondStockIn")
```

2.22:SetClose

原型：Boolean SetClose()

设置表单的关闭标志，在表单的入口事件中如果关闭标志被设置，那么表单将会被关闭。因此这个函数主要用于在入口事件中打开的其它表单用于控制父表单是否显示。

适用平台：All

参数：无

返回值：始终为True。

示例：

```
// 在StockInView的OnLoad事件是打开了CondStockIn

// 以下代码在CondStockIn中
// Cancel按钮的事件
SetClose()

// 这样如果在CondStockIn中按下了Cancel按钮，StockInView不会被显示，而是被关闭。
```

2.23:SetOID

原型：Boolean SetOID(OID)

设置表单的数据标识OID。

适用平台：All

参数：

• OID 表单的数据标识，Long或Integer型。

返回值：始终为True。

示例：

```
// 设置表单的OID为10001
SetOID(10001)
```

2.24:GetOID

原型：Long GetOID()

取得表单的数据标识OID。

适用平台：All

参数：无

返回值：表单的数据标识。

示例：

```
LoadData();
// 取得表单的数据标识
var OID = GetOID();
```

2.25:ApplyNewOID

原型：Long ApplyNewOID()

分配一个新的OID。

适用平台：All

参数：无

返回值：一个新的OID，Long型。

示例：

```
// 生成一个新的OID并输出
var newOID = ApplyNewOID();
OUT(newOID);
```

2.26:GetDataObjectKey

原型：String GetDataObjectKey(formKey)

取得指定表单关联的数据对象标识。

适用平台：All

参数：

- formKey 表单的标识，可选参数，如果未定义则取当前表单。

返回值：表单关联的数据对象标识。

示例：

```
// 取得表单关联的数据对象标识
var objKey = GetDataObjectKey();
```

2.27:GetFormKey

原型：String GetFormKey()

适用平台：All

参数：无

返回值：当前表单的表单标识。

示例：

```
// 取得当前表单的表单标识
var formKey = GetFormKey();
```

2.28:GetOperationState

原型：Integet GetOperationState()

取得表单的操作状态。使用频率不高的函数，主要用于字典树的刷新中判断如何刷新。

适用平台：All

参数：无

返回值：表单的操作状态。

示例：

```
// 取得表单的操作状态
var state = GetOperationState();
```

2.29:GetInitOperationState

原型：Integer GetInitOperationState()

取得表单的初始操作状态。使用频率不高的函数，主要用于字典树的刷新中判断如何刷新。

适用平台：All

参数：无

返回值：表单的初始操作状态。

示例：

```
// 取得表单的初始操作状态
var initState = GetInitOperationState();
```

2.30:Load

原型：void Load()

执行表单的载入操作。载入并显示之后需要计算未绑定数据源的组件和单元格的值；计算可见性、可用性。

适用平台：All

参数：无

返回值：无。

示例：

```
// 载入并显示10001的数据
SetOID(10001);
Load();
```

2.31:Save

原型：void Save()

执行表单的载入操作。保存之后表单转为默认状态。

适用平台：All

参数：无

返回值：无

示例：

```
// 执行保存操作
Save();
```

2.32:UpdateView

原型：Boolean UpdateView()

为表单更新其关联的表单列表视图的(序事簿)的内容，只有在表单的父表单的内容为View时执行更新，否则什么都不做。

适用平台：All

参数：无

返回值：始终返回True。

示例：

```
// 在表单的保存事件中
SaveData();
// 更新父界面的视图
UpdateView();
```

2.33:GetParentOID

原型：Long GetParentOID()

取得当前表单的父表单中的数据对象OID。

适用平台：All

参数：无

返回值：父表单的数据对象OID。

示例：

```
// 取得父表单的数据对象的OID
var parentOID = GetParentOID();
```

2.34:OpenDetail

原型：void OpenDetail(detailFormKey, OID, DOID)

打开一个明细表单。打开的明细表单可以保存至数据库。

适用平台：All

参数：

- detailFormKey 明细表单的标识；
- OID 数据对象的数据标识；
- DOID 数据对象的明细行的数据标识；

返回值：无

示例：

```
// 打开DetailStockIn明细表单
OpenDetail('DetailStockIn', OID, DOID)
```

2.35:OpenDetailInGrid

原型：void OpenDetailInGrid(detailFormKey, OID, DOID)

为表格中的明细行打开一个明细表单，在OID和DOID均不为-1的情况下，可以直接保存至数据库，也可以保存到父界面。这个函数通过记录父界面编辑的明细行的bookmark来记录数据行的关联。如果DOID为-1，为当前行打开明细表单。

适用平台：All

参数：

- detailFormKey 明细表单的标识；
- OID 数据对象的数据标识；

- DOID 数据对象的数据行的数据标识；

返回值：无

示例：

```
// 打开StockIn的明细的当前行
OpenDetailInGrid("DetailStockIn", GetOID(), -1)
```

2.36:NewDetail

原型：void NewDetail(detailFormKey, OID, VERID, DVERID)

使用明细表单新增一行明细。新增的明细表单可以保存到数据库。

适用平台：All

参数：

- detailFormKey 明细表单的标识；
- OID 数据对象的数据标识；
- VERID 数据对象的版本；
- DVERID 数据明细的明细控制版本；

返回值：无

示例：

```
// 使用明细表单DetailStockIn新增一行明细
NewDetail("DetailStockIn", OID)
```

2.37:NewDetailInGrid

原型：void NewDetailInGrid(detailFormKey, OID)

为表单中的表格新增一行。

适用平台：All

参数：

- detailFormKey 明细表单的标识；
- OID 数据对象的数据标识；

返回值：无

示例：

```
// 为StockIn的明细新增一行
NewDetailInGrid("DetailStockIn", GetOID())
```

2.38:EditDetail

原型：void EditDetail(detailFormKey, OID, DOID)

编辑一行明细。编辑的明细表单可以保存至数据库。

适用平台：All

参数：

- detailFormKey 明细表单的标识；
- OID 数据对象的数据标识；
- DOID 数据对象的明细行的数据标识；

返回值：无

示例：

```
// 编辑StockIn的明细表格的当前行
EditDetail("DetailStockIn", OID, DOID)
```

2.39:EditDetailInGrid

原型：void EditDetailInGrid(detailFormKey, OID, DOID)

编辑表单中表格的一行数据。在OID和DOID均不为-1的情况下，编辑的明细表单可以保存至数据库。在DOID为-1的情况下，编辑表格的当前行。

适用平台：All

参数：

- detailFormKey 明细表单的标识；
- OID 数据对象的数据标识；
- DOID 数据对象的明细行的数据标识；

返回值：无

示例：

```
// 编辑StockIn的当前行的数据
EditDetailInGrid("DetailStockIn", GetOID(), -1)
```

2.40:SaveDetail

原型：void SaveDetail(direct)

保存明细数据。保存有两种方式，一种是直接保存至数据库，另外一种是保存至父界面。只有在OID不为-1的情况下才可以直接保存至数据库。函数只能在明细表单中执行。

适用平台：All

参数：

- direct 是否直接保存至数据，默认值为false。

返回值：无

示例；

```
// 保存一行明细至数据库
SaveDetail(true)
```

第 3 章 界面处理函数

目录

3.1:SetValue

原型：Boolean SetValue(key, value, fireEvent)

给表单中的组件设置值，在需要触发值改变事件的情况下，执行值改变事件。

适用平台：All

参数：

- key 组件的标识
- value 新的值
- fireEvent 是否触发事件标志，可选参数，默认值为True。

返回值：在无异常的情况下，始终为True，否则无意义。

示例：

```
// 设置Amount的值为12.9
SetValue("Amount", 12.9)
```

3.2:GetValue

原型：Object GetValue(key)

返回组件的值，对于不同的组件，其返回值的类型根据不同的组件而定。

适用平台：All

参数：

- key 组件的标识

返回值：组件的值。

示例：

```
// 在数值编辑框中键入12.67
// 返回值为12.67
GetValue("Num")

SetValue("Memo", "test");
// 返回值为"test"
GetValue("Memo");
```

3.3:GetCellValue

原型：Object GetCellValue(key, row, col)

取得表格中指定单元格的值。

适用平台：All

参数：

- key 表格组件的标识
- row 行号，如果为-1取当前行。
- col 列。取值为整型或字符串类型，如果为字符串类型，则指的是明细单元格的标识。

返回值：单元格的值。

示例：

```
// 返回Detail表格当前行的Mtl字段的值
GetCellValue("Detail", -1, "Mtl")
```

3.4:SetCellValue

原型：Boolean SetCellValue(key, row, col, value, fireEvent)

设置表格中指定单元格的值，并可以指定是否触发值改变事件的处理。

适用平台：All

参数：

- key 表格组件的标识
- row 行号，如果为-1取当前行。
- col 列。取值为整型或字符串类型，如果为字符串类型，则指的是明细单元格的标识。
- value 新值。
- fireEvent 是否触发事件标志，可选参数，默认值为True。

示例：

```
// 设置Detail表格的当前行Mtl字段的值为10001
SetCellValue("Detail", -1, "Mtl", 10001);
```

3.5: RefreshControl

原型：void RefreshControl(ctrKey)

刷新控件,值刷新,重新展示

适用平台：All

参数：

- ctrKey 组件的标识

返回值：无

```
//重新刷新表格grid的值
RefreshControl("grid")
```

3.6:SetVisible

原型：void SetVisible(key, visible)

设置组件的可见性。

适用平台：All

参数：

- key 组件的标识
- visible 可见性，取值为True或者False。

返回值：无

```
// 设置组件"Memo"不可见
SetVisible("Memo", False)
```

3.7:SetEnable

原型：void SetEnable(key, enable)

设置组件的可用性。

适用平台：All

参数：

- key 组件的标识
- enable 可用性，取值为True或者False。

返回值：无

示例：

```
// 设置组件"Memo"不可编辑
SetEnable("Memo", False)
```

3.8:IsVisible

原型：Boolean IsVisible(key)

判断组件是否可见。

适用平台：All

参数：

- key 组件的标识

返回值：如果组件可见，返回True，否则返回False。

示例：

```
// 判断"Memo"是否可见
IsVisible("Memo")
```

3.9:IsEnable

原型：Boolean IsEnable(key)

判断组件是否可用。

适用平台：All

参数：

- key 组件的标识

返回值：如果组件可用，返回True，否则返回False。

示例：

```
// 判断"Memo"是否可用
IsEnable("Memo")
```

3.10:IsControlNull

原型：Boolean IsControlNull(key)

判断组件或者单元格是否为空值

适用平台：All

参数：

- key 组件或者单元格的标识

返回值：如果为空，返回true，否则返回false；

示例：

```
// 判断"test"是否为空值
IsControlNull("test")
```

3.11:SetFocus

原型：void SetFocus(key, focus)

设置组件的焦点。

适用平台：All

参数：

- key 组件的标识

- focus 焦点标志，取值为True或False。

返回值：无

示例：

```
// 设置"Memo"组件的焦点
SetFocus("Memo", True)
```

3.12:SetForeColor

原型：void SetForeColor(key, color)

给组件设置前景色。

适用平台：All

参数：

- key 组件的标识
- color 颜色，web颜色格式的字符串，比如#22ffcc或red。

返回值：无

示例：

```
// 设置前景色为红色
SetForeColor("Memo", "red")
```

3.13:SetBackColor

原型：void SetBackColor(key, color)

给组件设置背景色，只对面板有效。

适用平台：All

参数：

- key 组件的标识
- color 颜色，web颜色格式的字符串，比如#22ffcc或red。

返回值：无

示例：

```
// 设置面板的背景色为红色
SetBackColor("main", "red")
```

3.14:RunOpt

原型：void RunOpt(key)

执行指定标识的操作。

适用平台：PC

参数：

- key 操作的标识

返回值：无

示例：

```
// 执行标识为"Save"的操作
RunOpt("Save")
```

3.15:RunValueChange

原型：void RunValueChange(key)

执行组件或单元格的值改变事件。

适用平台：All

参数：

- key 组件或单元格的标识

返回值：无

示例：

```
// 执行"Memo"组件的值改变事件
RunValueChanged("Memo")
```

3.16:RunClick

原型：void RunClick(key)

执行按钮和超链接的点击事件。

适用平台：All

参数：

- key 按钮或超链接的标识

返回值：无

示例：

```
// 执行"OK"按钮的点击事件
RunClick("OK")
```

第 4 章 DictView界面操作函数

目录

4.1:ClearSelection

原型：Boolean ClearSelection(key)

清除DictView的选中行。

适用平台：All

参数：

- key DictView的标识

返回值：没有异常抛出的情况下，返回True，否则返回值无意义。

示例：

```
// 清除DictView的选中行
ClearSelection("dv")
```

4.2:GetSelectedValue

原型：Boolean GetSelectedValue(key, colKey)

获取dictview的选中行的对应列值。

适用平台：All

参数：

- key DictView的标识
- colKey 列的key

返回值：无异常抛出的情况下，返回True，否则返回值无意义。

示例：

```
// DictView的选中行上列(key为name)的值
GetSelectedValue("dv", "name")
```

第 5 章 字典界面操作函数

目录

5.1:NewDict

原型：Boolean NewDict(formKey)

新建一个字典。如果当前表单没有内嵌的默认容器，则在新的tab页上打开，否则在表单内嵌的默认容器中打开。

适用平台：All

参数：

- formKey 字典表单的标识。

返回值：没有异常抛出的情况下，返回True，否则返回值无意义。

示例：

```
// 打开一个物料字典
NewDict("Mtl")
```

5.2:LoadDict

原型：Boolean LoadDict(OID)

在表单上打开一个字典的数据。

适用平台：All

参数：

- OID 字典的数据对象OID

返回值：无异常抛出的情况下，返回True，否则返回值无意义。

示例：

```
// 打开数据对象标识为10001的字典的数据
LoadDict(10001)
```

5.3:OpenDict

原型：Boolean OpenDict(formKey, OID)

通过字典的标识和数据对象OID打开一个字典。默认打开在当前表单的默认容器，如果没有默认容器，打开在新tab页中。

适用平台：All

参数：

- formKey 字典的标识
- 字典数据对象OID

返回值：没有异常抛出的情况下，返回True，否则返回值无意义。

示例：

```
// 打开字典
OpenDict('Material', 10001)
```

5.4:GetDictValue

原型：Object GetDictValue(ItemKey, OID, FieldKey)

获取字典项中某个属性的值。

适用平台：All

参数：

- ItemKey 字典标识；
- OID 字典项的对象标识；
- FieldKey 属性的标识，以"表名.列名"的形式定义。

返回值：字典项的属性值。

示例：

```
// 取得字典项的代码
GetDictValue("Material", 10001, "MaterialHead.Code")
```

第 6 章 表格函数

目录

6.1:InsertRow

原型：Integer InsertRow(key, row)

在表格上插入新行。key指定了需要插入行的表格，row为插入位置，如果row未定义，在当前位置插入行；如果row为-1，在表格最后插入一行。

适用平台：All

参数：

- key 表格组件的标识
- row 插入行的位置，可选参数。默认值为表格的当前行。

返回值：新增行的序号。

示例：

```
// 向Detail表格的当前行位置插入一新行
var newRow = InsertRow("Detail");
```

6.2:DeleteRow

原型：Boolean DeleteRow(key, row)

删除表格中的指定行。

适用平台：All

参数：

- key 表格组件的标识
- row 行号，可选参数，默认值为表格的当前行。

返回值：没有抛出异常的情况下返回True，否则返回值没有意义。

示例：

```
// 删除Detail表格的第2行
DeleteRow("Detail", 2)
```

6.3: ReloadTable

原型：void ReloadTable(dataTableKey)

重新加载Document中的Datatable,只载入特定的表,其他表不载入

适用平台：PC

参数：

- key 数据源表的标识

返回值：没有抛出异常的情况下返回True，否则返回值没有意义。

示例：

```
// 重新加载数据源表table1
ReloadTable("table1")
```

6.4: ReplaceTable

原型：void ReplaceTable(dataTableKey,dataTable)

替换Document中指定的DataTable,

适用平台：PC

参数：

- dataTableKey 需要被替换的DataTable的key
- dataTable 替换的新的DataTable

返回值：没有抛出异常的情况下返回True，否则返回值没有意义。

示例：

```
// 替换源数据表table1为新的DataTable
ReplaceTable("table1",DataTable)
```

6.5:Sum

原型：BigDecimal Sum(cellKey)

统计cellKey的汇总值。

适用平台：All

参数：

- cellKey 单元格的标识

返回值：行作用域内(分组内)相同标识单元格的值的合计，BigDecimal类型。

示例：

```
// 计算Num的汇总值
var sumValue = Sum("Num");
```

6.6:SumExpand

原型：BigDecimal SumExpand(cellKey)

统计当前行中列相同扩展单元格的合计值。

适用平台：PC

参数：

- cellKey 单元格的标识

返回值：当前行内相同列扩展单元格的合计值。BigDecimal类型。

示例：

```
// 列扩展根据颜色(红、绿、蓝扩展)，计算当前行的"Num"的汇总值
var expandSumValue = SumExpand("Num");
```

6.7: CopyGridRow

原型：Integer CopyGridRow(gridKey, rowIndex, splitKeys, value1,... ,valueN)

复制被复制的行形成新行插入到该行下方，排除需要排除的列的值。

适用平台：PC

参数：

- gridKey 表示表格组件的标识
- rowIndex 表示要复制的行的序号。如果为-1，则表示复制当前选中行。
- splitKeys 表示不需要复制值的列的标识，格式为：“key1,key2,key3,... ”
- value1,...,valueN 表示不需要复制的列的填入的新的值，顺序同splitKeys

返回值：复制出来的新行的序号。Integer类型。

示例：

```
// 将当前选中的行进行复制,不复制地区(Area)和区号(AreaNum)的值，填入新值“上海”，“021”
var newRowIndex = CopyGridRow("detail_grid",-1,"Area,AreaNum","上海","021");
```

6.8: GetRowCount

原型：Integer GetRowCount(key,allRow)

取得ListView和Grid组件的行数

适用平台：PC

参数：

- key 网格组件的标识
- allRow 可选参数,是否包含空白行,对表格有效,默认不包含

返回值：行数。Integer类型。

示例：

```
// 获取表格（detail_grid）的行数
var rowCount = GetRowCount("detail_grid");
```

6.9: SetRowIndex

原型：void SetRowIndex(key,rowIndex)

设置公式计算过程中集合类组件的当前行

适用平台：PC

参数：

- key ListView和Grid组件的标识
- rowIndex 行序号

返回值：无。

示例：

```
// 设置表格（"detail_grid"）的当前行为10
SetRowIndex("detail_grid",10);
```

6.10: CheckDuplicate

原型：boolean CheckDuplicate(cellKey)

判断表格一列单元格的值是否重复,不支持有拓展的表格

适用平台：PC

参数：

- cellKey 单元格的标识

返回值：是否重复。Boolean类型

示例：

```
// 判断表格("detail_grid")的列地区（Area）是否有值重复
var isDuplicate = CheckDuplicate("Area");
```

6.11:IsEmptyRow

原型：boolean IsEmptyRow(gridKey,rowIndex)

判断表格某行是否为空白编辑行

适用平台：PC

参数：

- gridKey 表格组件标识
- rowIndex 行序号

返回值：是否为空白编辑行。Boolean类型

示例：

```
// 判断表格("detail_grid")的当前行是否为空白编辑行
var isEmpty = IsEmptyRow("detail_grid",-1);
```

6.12: ClearAllRows

原型：void ClearAllRows(gridKey)

清除表格的所有数据行,保留空白编辑行

适用平台：PC

参数：

- gridKey 表格组件标识

返回值：无

示例：

```
// 清空表格("detail_grid")的数据行，保留空白编辑行
ClearAllRows("detail_grid");
```

6.13:SetColumnVisible

原型：SetColumnVisible(gridKey,columnKey,visible)

设置表格列的可见性

适用平台：PC

参数：

- gridKey 表格组件标识
- columnKey 列标识
- visible 是否可见

返回值：无

示例：

```
// 设置某一行可见
SetColumnKey('detail_grid','Amount',true);
```

6.14:FillGrid

原型：FillGrid(gridKey,datatable,cellkey1,cellkey2,.....)

填充表格数据

适用平台：PC

参数：

- gridKey 表格组件标识

- datatable 数据集
- cellkey1 按顺序设置需要填充的单元格标志1
- cellkey2 按顺序设置需要填充的单元格标志2
- cellkeyN ......

返回值：无

示例：

```
// 填充表格数据
FillGrid({detail_grid},DBNamedQuery({MaterialQuery}, MaterialCond),{Material},{Area},{Amount});
```

第 7 章 认证相关函数

目录

7.1:Login

原型：Boolean Login(user, password, role, clusterID)

用户登录函数。

适用平台：All

参数：

- user 用户代码
- password 密码
- role 角色，-1时为该用户的所有角色。可选参数，未定义是为-1。
- clusterID 数据化数据标识，可选参数，默认值为-1。

返回值：登录成功返回True，否则抛出异常。

示例：

```
// 以用户user，密码11111登录
Login("user", "111111")
```

7.2:Logout

原型：Boolean Logout()

用户登出。

适用平台：All

参数：无

返回值：无异常的情况下返回True。

示例：

```
// 用户登出
Logout()
```

7.3:GetOperator

原型：Long GetOperator()

取得当前登录用户标识。

适用平台：All

参数：无

返回值：当前登录用户的标识。Long类型。

示例：

```
// 取得当前登录用户标识并存储在变量operator中
var operator = GetOperator();
```

第 8 章 界面数据处理函数

目录

8.1:DBUpdate

原型：void DBUpdate(sql, params...)

使用sql更新数据。

适用平台：All

安全等级要求：10以下

参数：

- sql 更新语句
- params 不定长参数，为sql语句中的?参数

返回值：无

示例：

```
// 更新StockInHead的数据
DBUpdate("update StockInHead set Mtl=?,Num=? where OID=?", 10001, 13, 20001)
```

8.2:DBQuery

原型：DataTable DBQuery(sql, params...)

使用sql语句查询数据并返回DataTable。

适用平台：All

安全等级要求：10以下

参数：

- sql 查询语句
- params 不定长参数，为sql语句中的?参数

返回值：DataTable。

说明:根据实际传入的参数类型为SQL语句设置参数值,需要自行保持类型一致。

示例：

```
// 查询StockInHead中Mtl为10001的数据
var table = DBQuery("select * from StockInHead where Mtl=?", 10001)
```

8.3:DBQueryValue

原型：Object DBQueryValue(sql, params...)

使用sql语句查询数据，并返回结果集的第一行第一列的数据。

适用平台：All

安全等级要求：10以下

参数：

- sql 查询语句
- params 不定长参数，为sql语句中的?参数

返回值：Object，为查询结果集中第一行和第一列的数据，如果不存在相应的数据，返回null。

说明:根据实际传入的参数类型为SQL语句设置参数值,需要自行保持类型一致。

示例；

```
// 查询StockInHead中Mtl为10001的数据行数
var count = DBQueryValue("select count(*) from StockInHead where Mtl=?", 10001)
```

8.4:DBNamedQuery

原型：DataTable DBNamedQuery(key, params...)

根据中间层指定名称的sql语句查询结果集。

适用平台：All

参数：

- key 中间层预定义查询语句的标识
- params 不定长参数，为查询语句中?参数的值

返回值：DataTable。

说明 ： 如果后台的查询预定义中没有定义查询参数的DataType，那么根据实际传入的参数类型为SQL语句设置参数.需自行控制好类型一致。

预定义查询的位置为当前表单或者是CommonDef中。

示例；

```
// 后台有一个预定义查询语句QueryStockInList，其sql语句为"select * from StockInHead where Mtl=?"
<QueryCollection>
  <Query Key="QueryStockInList" Description="测试用" Args="">
   <Statement>
    <![CDATA[
     select * from StockInHead where Mtl=?
    ]]>
   </Statement>
   <ParameterCollection>
    <Parameter DataType="Long"/>
   </ParameterCollection>
  </Query>
```

```
</QueryCollection>

//前台定义为:
var table = DBNamedQuery("QueryStockInList", 10001)
```

8.5:DBNamedQueryValue

原型：Object DBNamedQueryValue(key, params...)

根据中间层指定名称的sql语句查询结果集，并返回第一行第一列的值。

适用平台：All

参数：

- key 中间层预定义查询语句的标识
- params 不定长参数，为查询语句中?参数的值

返回值：查询结果集中第一行第一列的值，没有相应的值，返回null。

说明 ： 如果后台的查询预定义中没有定义查询参数的DataType，那么根据实际传入的参数类型为SQL语句设置参数.需自行控制好类型一致。

预定义查询的位置为当前表单或者是CommonDef中。

示例：

```
// 后台有一个预定义查询语句QueryCount，其sql语句为"select count(*) from StockInHead where Mtl=?"
<QueryCollection>
  <Query Key="QueryCount" Description="测试用" Args="">
   <Statement>
    <![CDATA[
     select * from StockInHead where Mtl=?
    ]]>
   </Statement>
   <ParameterCollection>
    <Parameter DataType="Long"/>
   </ParameterCollection>
  </Query>
</QueryCollection>

//前台定义为:
var count = DBNamedQueryValue("QueryCount", 10001)
```

8.6:ServerDate

原型：DateTime ServerDate()

返回中间层的服务器时间。

适用平台：All

参数：无

返回值：中间层服务器时间。

示例：

```
// 返回服务器时间
var t = ServerDate()
```

8.7:ServerDBDate

原型：DateTime ServerDBDate()

返回中间层数据库时间。

适用平台：All

参数：无

返回值：中间层数据库时间。

示例：

```
// 返回数据库时间
var t = ServerDBDate()
```

8.8:ReloadTable

原型：void ReloadTable(key)

重载指定数据表的数据。

适用平台：All

参数：

- key 数据源中表的标识

返回值：无

示例：

```
// 重载"Detail"表的数据
ReloadTable("Detail")
```

8.9:RefreshControl

原型：void RefreshControl(key)

根据文本中的数据刷新表格。

适用平台：All

参数：

- key 表格组件的标识，包括ListView和Grid。

返回值：无

```
// 刷新"Detail"表格控件的内容
RefreshControl("Detail")
```

8.10:ReplaceTable

原型：void ReplaceTable(key, table)

替换数据源中表的数据。用table的数据替换原有的key所在的表的数据。

适用平台：All

参数：

- key 数据源表的标识

- table 数据表DataTable。

返回值：无

示例：

```
// 替换"Detail"表的数据
var t = DBQuery("select * from StockInHead");
ReplaceTable("Detail", t);
```

第 9 章 表单文档函数

目录

9.1:GetStatus

原型：int GetStatus()

功能：取得表单文档的状态值。

参数列表：无

返回值：如果表单定义的状态字段，则返回状态的值，否则抛出异常。

示例：

```
// 取得表单的文档状态
var status = GetStatus();
```

第 3 部分 中间层函数参数

目录

第 10 章 中间层基础函数

目录

10.1:SetValue

原型：void SetValue(tableKey, columnKey, value)

给表的某个字段设置值；

适用平台：N/A

参数：

- tableKey 表的标识；
- columnKey 列的标识
- value 新值

返回值：无

示例：

```
// 设置StockInHead表的Mtl字段的值为10001
SetValue("StockInHead", "Mtl", 10001)
```

第 4 部分 业务蓝图

目录

第 11 章 业务流程客户端函数

目录

11.1:StartInstance

原型：void StartInstance(processKey)

为数据对象启动流程实例，在指定processKey的情况下启动processKey的实例，如果processKey为空，通过中间层的的表单和过程关联查询过程标识。

适用平台：All

参数：

- processKey 过程的标识，可以为空。

返回值：无

示例：

```
// 启动请假流程
StartInstance("VacationRequest")
```

11.2:OpenWorkitem

原型：void OpenWorkitem(workitemID, shell)

在客户端打开工作项，这个过程通常打开工作项所在实例的关联表单，在界面上添加工作流相关操作。

适用平台：All

参数：

- workitemID 工作项的标识；
- shell 工作项打开的表单的外壳程序；TODO:具体的形式暂时未定，待实现后再作统一规定。

返回值：无

示例：

```
// 打开待办中当前行的工作项，这里假定待办中有WorkitemID这一列
OpenWorkitem(WorkitemID)
```

11.3:AuditWorkitem

原型：void AuditWorkitem(workitemID, result, userInfo)

完成一个审批工作项，用于对一个审批任务进行处理，需要指定工作项标识，审批结果和用户意见。

适用平台：All

参数：

- workitemID 工作项的标识；为-1时表示从当前界面取得活动工作项。
- result 审批结果，通常情况下0为驳回，1为同意，其它值代表扩展选项。
- userInfo 为用户的意见，通常为审批者的录入文本。

返回值：无

示例：

```
// 同意当前活动审批工作项
AuditWorkitem(-1, 1, "同意，请酌情办理！")
```

11.4:CommitWorkitem

原型：void CommitWorkitem(workitemID, result, userInfo)

提交一个工作项，用于结束一个工作项。

适用平台：All

参数：

- workitemID 工作项的标识；为-1时表示从当前界面取得活动工作项；
- result 处理结果；
- userInfo 为用户的处理意义，通常为处理者的录入文本。

返回值：无

示例：

```
// 提交当前活动工作项
CommitWorkitem(workitemID, -1, "已处理")
```

11.5:BPMMap

原型：void BPMMap(mapKey)

执行业务过程中的数据映射，同普通的映射不同的地方在于，需要携带映射任务中的工作项信息。

适用平台：PC

参数：

- mapKey 数据映射的标识；

返回值：无

示例：

```
// 执行订单映射入库
BPMMap("Order2StockIn")
```

11.6:GetProcessKey

原型：String GetProcessKey()

取得表单中的使用的过程标识，如果当前表单已经有流程实例关联，取得的是使用的过程的标识，如果当前表单没有流程实例关联，取得的是表单(或数据对象)的关联过程标识。

适用平台：All

参数：无

返回值：如果有关联的过程，返回其过程标识，否则返回""。

示例：

```
// 取得表单中使用的过程标识
var key = GetProcessKey();
```

11.7:GetProcessVer

原型：Integer GetProcessVer()

取得表单中使用的过程的版本，如果当前表单已经有流程实例关联，取得的是使用的过程的版本，如果当前表单没有流程实例关联，取得的是表单(或数据对象)的关联过程版本。

适用平台：All

参数：无

返回值：如果有关联的过程，返回其过程版本，否则返回-1。

示例：

```
// 取得表单中使用的过程版本
var ver = GetProcessVer();
```

第 5 部分 工程及应用属性参考

目录

第 12 章 工程及应用

目录

12.1:应用

12.1.1:结构

```
<Root>
    - Solution.xml
    - setting.xml 应用设置文件
    CommonDef.xml - 应用公共定义，具体定义详见项目中的定义
    AndroidDef.xml - 安卓手机及平板的定义
    <Resource>
        ...
    <Data> - 应用的数据存储目录(一般为附件)
        ...
    <Project>
    ...
```

- <Root>为应用的根目录；
- Solution.xml 为应用的定义文件；
- setting.xml 为应用的运行设计文件；
- CommonDef.xml 为公共定义文件，在应用与工程之间具有不同的应用优先级；
- AndroidDef.xml 为Android应用的定义；
- <Resource> 目录为应用的资源；
- <Data> 应用的数据存储目录(一般为附件)；
- <Project> 为工程的路径，一个应用包含一个或多个工程目录；

12.1.2:基础定义说明

这里的基础定义部分包含了系统用到的一些通用定义，这些定义在不同的地方被重复说明，因此在这里说明。

12.1.2.1:脚本

这里的脚本说明了脚本类型的属性，但不表示xml节点的名称，节点的名称在具体定义的地方会被替换。

所有的表达式、JavaScript和Java代码的公共组成部分。格式如下：

```
<Tag Type="" RunType="">
        <![CDATA[
...
        ]]
</Tag>
```

- Tag 标记在具体的属性定义中会被具体的类型替代，这里只是一个节点的抽象标记；

- Type 脚本的类型，取值如下：
  - formula 表达式；
  - javascript JavaScript脚本；
  - java Java类；
- RunType 脚本的执行类型，用于定义脚本的执行环境，取值如下：
  - Undefined 无定义，表示无限制；
  - Client 只在客户端执行，针对web环境；
  - Server 只在服务端执行。

  默认值为Undefined。

12.1.2.2:尺寸定义

尺寸定义以字符串形式表示，有以下几种定义形式：

- 固定像素，定义形式为以px结尾的定义，比如10px，表示10像素；
- 固定Android平台像素，定义形式为以apx结尾的定义，比如10apx，表示10个Android像素；
- 比例，定义形式为以%结尾的定义，比如50%；
- 内容自适应，定义形式为pref；
- 固定逻辑单位，为Android所使用，为以dp结尾的定义，比如10dp；
- 自动分配，定义形式为auto；这种定义表示以设备默认形式来分配。

12.1.2.3:参数集定义

```
<ParameterCollection>
    <Parameter TargetTable="" TargetColumn="" DataType="" SourceType="" Formula="" FieldKey="" Value="" Description=""/>
    ...
</ParameterCollection>
```

Parameter属性

- TargetTable 参数的目标表，当前未使用；
- TargetColumn 参数的目标字段；可以不定义，不定义的情况下，根据DataType定义来确定参数的类型，如果在DataType也未定义的情况下，根据实际值来确定参数类型；
- DataType 定义参数的类型，在定义了TargetColumn的情况下，通过列的定义推算。
- SourceType 参数的来源类型，取值范围为Formula(来源于公式)、Field(来源于字段)、Const(常量)，默认值为Formula；
- Formula 在SourceType为Formula时定义表达式；
- FieldKey 在SourceType为Field时定义来源字段(组件标识或单元格标识)；
- Value 在SourceType为Const时定义常量值。
- Description 描述。

12.1.3:Solution.xml文件定义

```
<Solution>
```

```
    <!-- 工程集合定义 -->
    <ProjectCollection>
        <Project>
        ...
    </ProjectCollection>
    <!-- 工具栏扩展定义 -->
    <InplaceToolbarCollection>
        <InplaceToolbar/>
        ...
    </InplaceToolbarCollection>
    <!-- 中间层处理流扩展定义 -->
    <MidProcessFlowCollection>
        <MidProcessFlow Service="">
            <MidProcess/>
            ...
        </MidProcessFlow>
    </MidProcessFlowCollection>
    <!-- 业务蓝图运行设计 -->
    <BPMSetting>
        <ClientBPMAction>
        </ClientBPMAction>
    </BPMSetting>
    <!-- 设备平台相关初始启动项定义 -->
    <StartItemCollection>
        <StartItem/>
        ...
    </StartItemCollection>
</Solution>
```

12.1.3.1:Solution属性

```
<Solution Key="" Caption="" StartForm="" SecurityLevel="" DataPath=""/>
```

- Key 应用的标识；
- Caption 应用的名称；
- StartForm 启动表单；
- SecurityLevel 安全等级定义，不同的安全等级定义会限制客户端对中间层的调用；
- DataPath 附件的存储目录定义；

12.1.3.2:ProjectCollection工程集合

```
<ProjectCollection>
    <Project Key="" Caption=""/>
    ...
</ProjectCollection>
```

- Key 工程的标识；
- Caption 工程的名称；

12.1.3.3:InplaceToolbarCollection工具栏扩展集合定义

```
<InplaceToolbarCollection>
    <InplaceToolbar Tag="" Handler="" Description=""/>
    ...
</InplaceToolbarCollection>
```

工具栏扩展集合定义包含多个扩展，扩展的属性如下：

- Tag 定义在表单操作中的处理标记；
- Handler - 处理类全路径名，类必须实现com.bokesoft.yes.view.model.IInplaceToolBarCallback接口；
- Description 描述；

12.1.3.4:MidProcessFlowCollection中间层处理流集合定义

```
<MidProcessFlowCollection>
    <MidProcessFlow Service="">
        <MidProcess Description="" Time="" Impl=""/>
        ...
    </MidProcessFlow>
</MidProcessFlowCollection>
```

中间层处理流用于对中间层服务进行扩展，其包含对多个服务的扩展，每个服务包含对不同时机的扩展。

服务扩展属性如下：

- Service 定义服务的名称；

服务时岂的扩展属性如下：

- Description 处理的描述；
- Time 时机，取值为Pre或Post，默认值为空，Pre表示服务执行前的扩展，Post表示服务执行后的扩展；
- Impl 扩展的实现的全路径名，必须为实现com.bokesoft.yes.mid.service.IServiceProcess<T extends IServiceContext>接口的类；

12.1.3.5:BPMSetting设置

```
<BPMSetting>
    <ClientBPMAction MobileDefaultWorkItemAction="">
    </ClientBPMAction>
</BPMSetting>
```

保留部分，暂时未作定义。

12.1.3.6:StartItemCollection启动项集合定义

```
<StartItemCollection>
    <StartItem Platform="" StartForm=""/>
    ...
</StartItemCollection>
```

启动项集合定义包含多个启动项，每个启动项的属性如下：

- Platform 设备平台，取值包括：All、Android、后续待扩充部分。
- StartForm 启动表单的标识；

12.1.4:setting.xml文件

12.1.4.1:结构

```
<Setting>
    <!-- 附件服务设置 -->
    <AttachmentService>
    </AttachmentService>
    <!-- 会话设置 -->
    <Session>
    </Session>
    <!-- 中间层启动监听 -->
    <MidListener>
    </MidListener>
    <!-- 皮肤设置 -->
    <Skin/>
</Setting>
```

- AttachmentService 附件服务设置，目前预留；
- Session 会话设置；
- MidListener 中间层启动服务监听，目前预留；
- Skin 皮肤。

12.1.4.2:Session会话设置

```
<Session Timeout="" EnableLock="" MaxLoginCount="">
</Session>
```

- Timeout为超时时间，单位为秒。默认为1小时，即3600秒；
- EnableLock 是否启用锁定，即不成功登录超出次数后锁定登录用户；
- MaxLoginCount 最大登录次数，即一个用户自上次成功登录以下不成功的登录次数。

12.1.4.3:MidListener中间层启动服务监听

```
<MidListener>
    <MidListenerTask>
        <![CDATA[...]]
    </MidListenerTask>
    ...
</MidListener>
```

启动服务包含多个监听任务，每个任务为一个脚本定义(脚本定义见脚本说明)。在Type为Java时，实现类必须实现com.bokesoft.yes.base.IStartListener接口；

12.1.4.4:Skin皮肤设置

```
<Skin skin=""/>
```

skin为皮肤标识。

12.2:CommonDef.xml公共定义

12.2.1:结构

```
<CommonDef>
    <!-- 操作定义集合 -->
    <OperationCollection>
    </OperationCollection>
    <!-- 状态定义集合 -->
    <StatusCollection>
    </StatusCollection>
    <!-- 脚本定义集合 -->
    <ScriptCollection>
    </ScriptCollection>
    <!-- 宏公式定义集合 -->
    <MacroCollection>
    </MacroCollection>
    <!-- 侧边栏菜单定义 -->
    <SlidingMenuDefCollection>
    </SlidingMenuDefCollection>
    <!-- 参数表 -->
    <ParaTable>
    </ParaTable>
    <!-- 查询集合 -->
    <QueryCollection>
    </QueryCollection>
</CommonDef>
```

- OperationCollection 操作定义集合，定义应用、工程或表单范围内的操作；

- StatusCollection 状态定义集合，定义应用、工程或表单范围内的状态；
- ScriptCollection 脚本定义集合，定义应用、工程或表单范围内的预定义脚本；
- MacroCollection 宏公式集合，定义应用、工程或表单范围内的宏公式；
- SlidingMenuDefCollection 侧边栏菜单定义，只在应用中定义；
- ParaTable 参数表，只在应用中定义；

12.2.2:OperationCollection操作集合

```
<OperationCollection>
    <Operation RefKey="" Key="" Caption="" Icon="" Enable="" Visible="" Tag="" SelfDisable="" Expand="" ExpandSource=""
 ShortCuts="">
        <Action>
            <![CDATA[  ]]
        </Action>
    </Operation>
    ...
    <OperationCollection>
        ...
    </OperationCollection>
    ...
</OperationCollection>
```

操作集合包含多个操作定义，也可以嵌套操作集合，只支持一层嵌套。操作的属性如下：

- RefKey 引用的上层定义的标识，在不为空的情况下用于引用上层定义操作；
- Key 操作的标识；
- Caption 操作的名称；
- Icon 图标定义；
- Enable 可用性，表达式；
- Visible 可见性，表达式；
- Tag 用于扩展工具栏的标记；
- SelfDisable 在包含操作集合的情况下，用于说明自身是否禁用；
- Expand 是否是扩展操作，取值为True和False，默认为False；
- ExpandSource 在Expand为True的情况下，定义扩展的源，表达式，返回值为"tag1,caption1;tag2,caption2..."的形式的字符串；
- ShortCuts 快捷键；
- Action 为操作执行的内容，"脚本"类型；

12.2.3:StatusCollection状态集合

```
<StatusCollection>
    <Status Key="" Caption="" Value="" Standalone="">
    </Status>
    ...
</StatusCollection>
```

状态集合由多个状态组成，状态集合的定义在应用、工程或表单层次。状态的属性如下：

- Key 状态的标识；
- Caption 状态的名称；

- Value 状态的值；
- Standalone 是否独立状态，在表单中该属性有效，表示定义自己独立的状态定义，即使在全局定义中存在，也使用自身的属性。

12.2.4:ScriptCollection脚本定义集合

```
<ScriptCollection>
    <Script Key="" Caption="" Description="" Range="" Verb="">
        <![CDATA[ ]]
    </Script>
    ...
</ScriptCollection>
```

脚本定义集合在应用、工程或表单层次定义预定义脚本，包含多个脚本定义。脚本为"脚本"类型定义，除了具体"脚本"的属性外，还具有如下属性：

- Key 标识；
- Caption 名称；
- Description 描述；
- Range 作用范围，取值如下：
  - Rule 规则；
  - Action 动作；
  - All 所有范围；

  默认值为All。
- Verb 脚本的用处，取值如下：
  - Load 用于表单载入；
  - Save 用表表单保存；
  - Other 其它；

  默认值为Other。

12.2.5:MacroCollection宏公式定义集合

```
<MacroCollection>
    <Macro Key="" Args="">
        <![CDATA[ ]]
    <Macro/>
    ...
</MacroCollection>
```

宏公式定义集合包含多个宏定义，宏定义的属性如下：

- Key 宏的标识，在表达式中引用此标识引用宏公式；
- Args 宏公式的参数，以逗号分隔的字符串，宏公式内部使用这些参数来引用外部传入的参数值；
- CDATA部分为宏公式的内容；

12.2.6:SlidingMenuDefCollection侧边栏菜单

侧边栏菜单为移动专用定义。这里暂时不用说明。

12.2.7:ParaTable参数表

```
<ParaTable>
    <ParaGroup Key="" Caption="">
        <ParaItem Key="" Caption="" Value=""/>
        ...
    </ParaGroup>
    ...
</ParaTable>
```

参数表由多个分组组成，分组属性如下：

- Key 分组标识；
- Caption 分组名称；

每个分组由多个参数项组成，参数项属性如下：

- Key 参数标识；
- Caption 参数名称；
- Value 参数值；

12.2.8:QueryCollection查询集合

```
<QueryCollection>
    <Query Key="" Description="" Args="">
        <![CDATA[...]]
    </Query>
    ...
</QueryCollection>
```

查询集合由多个查询组成，每个查询属性如下：

- Key 查询的标识，通过此标识调用查询；
- Description 查询的描述；
- Args 查询的参数；
- CDATA 部分为查询的内容，参数在其中以?表示；

12.3:工程

12.3.1:结构

```
<Root>
    - Project.xml - 工程描述文件
    - CommonDef.xml - 项目公共定义
    - Schema.xml - 数据库定义集合
    - <Schema> - 数据库定义存储目录
        - Table1.xml - 具体的表的描述文件
    - <DataObject> - 数据库对象存储目录
        <Sub1>
            DataObject1.xml - 具体的数据对象的描述文件
            ...
        <Sub2>
        - DataObject2.xml - 具体的数据对象的描述文件
        ...
    - <DataMap> - 数据映射存储目录
        <Sub1>
            DataMap1.xml
        ...
    - <DataMigration> - 数据迁移存储目录
        <Sub1>
            DataMigration1.xml
        ...
```

```
- <Form> - 窗口对象存储目录
    <Sub1>
        Form1.xml - 具体的窗口对象的描述文件
        ...
    <Sub2>
    - Form2.xml - 具体的窗口对象的描述文件
    ...
- <Template> -- 模板存储路径
    ...
- <Report> - 报表的存储目录
    ...
- BPM.xml - 流程布署信息和与表单的关联信息
- <BPM> - 流程对象存储目录
    - BPM1.xml - 具体的流程对象的描述文件
- Entry.xml - 功能入口定义描述
```

- <Root>为工程的根目录；
- Project.xml 工程描述文件；
- CommonDef.xml，工程中的公共设置，定义同应用中的同名文件，优先级高于应用下的同名定义；
- Schema.xml，工程的数据描述文件，目前没有使用，保留；
- <Schema>目录，定义工程的数据描述文件集合，目前没有使用，保留；
- <DataObject>目录，数据对象定义存储目录，可包含子目录；
- <DataMap>目录，数据映射关系存储目录，可包含子目录；
- <DataMigration>目录，迁移关系存储目录，可包含子目录；
- <Form>目录，表单定义存储目录，可包含子目录；
- BPM.xml，工程中流程对象定义集合文件；定义工程中流程布署信息和与表单的关联信息；
- <BPM>目录，流程定义存储目录，可包含子目录；
- Entry.xml，功能入口定义文件；

12.3.2:Entry.xml入口定义

```
<Entry Key="" Caption="" Visible="" Open="" Style="">
    <EntryItem Key="" ShortKeys="" Caption="" Type="" FormKey="" Enable="" Visible="" Parameters="" Layout="" Media=""
 Size="">
        <Action>
            <![CDATA[ ]]
        </Action>
    </EntryItem>
    ...
    <Entry Key="" Caption="" Visible="" Open="">
        ...
    </Entry>
    ...
</Entry>
```

Entry属性如下：

- Key 标识；
- Caption 名称；
- Visible 可见性；
- Open 是否展开；
- Style 样式，只在根层有效；取值为：Tree(树形)、GroupTree(分组树型)；

EntryItem属性如下：

- Key - 入口的标识，要求在同一个子树下不重复。
- ShortKeys - 快捷操作键，以逗号(,)分隔支持多个快捷键。
- Caption - 入口的名称。
- Type - 入口的类型，取值如下：取值为表单(Form)、表达式(Script)。
  - Form 表单；
  - Script 脚本；
  - URL 链接；
- FormKey - 打开的表单的标识。
- Action - 入口的操作。取值为Formula时为表达式；取值为Custom为定制的js函数名或定制的java类名称。
- Enable - 可用性。
- Visible - 可见性。
- Parameters - 初始参数集，以key=value形式表示，如果有多个参数，参数之间以分号隔开。
- Open 是否展开，Entry有些属性，EntryItem没有。
- Layout 表单的布局；
- Media 模拟的媒体，测试用；
- Size 模拟的界面大小，测试用。

第 6 部分 数据模型属性参考

目录

第 13 章 数据对象属性

目录

13.1:数据对象结构

```
<DataObject>
    <!-- OID过滤条件 -->
    <OIDFilter>
    </OIDFilter>
    <!-- 数据表集合 -->
    <TableCollection>
    </TableCollection>
    <!-- 关系定义 -->
    <Relation>
    </Relation>
    <!-- 检查规则集合 -->
    <CheckRuleCollection>
    </CheckRuleCollection>
    <!-- 载入前处理定义 -->
    <PreLoadProcess>
    </PreLoadProcess>
    <!-- 载入后处理定义 -->
    <PostLoadProcess>
    </PostLoadProcess>
    <!-- 保存前处理定义 -->
    <PreSaveProcess>
    </PreSaveProcess>
    <!-- 保存后处理定义 -->
    <PostSaveProcess>
    </PostSaveProcess>
    <!-- 中间层二次开发扩展定义 -->
    <ExtendCollection>
    </ExtendCollection>
</DataObject>
```

说明：

- TableCollection 表集合定义，数据对象的主体由多个数据表组成，这里定义表集合；
- Relation 定义多个数据对象之间的层次关系，定义好的关系数据对象由复合字典使用；
- CheckRuleCollection 检查规则集合，定义数据对象必须满足的条件；
- PreLoadProcess 载入前的处理，在数据载入之前做的额外操作定义；
- PostLoadProcess 载入后的处理，在数据载入后做的额外操作定义；
- PreSaveProcess 保存前处理，在数据保存前做的额外操作定义，主要用于对数据进行再次加工；
- PostSaveProcess 保存后处理，在数据保存后做的额外操作定义，主要用于不会修改当前表单数据的处理，比如发通知等。
- ExtendCollection 中间层二次开发扩展定义，用于引入二次开发的类，这些类可以在表达式中调用。

13.2:数据对象属性

```
<DataObject Key="" Caption="" PrimaryType="" SecondaryType="" PrimaryTableKey="" NoPrefix=""/>
```

- Key 数据对象标识，全局唯一；
- Caption 数据对象名称，对数据对象的简要说明；
- PrimaryType 数据对象的主类型，取值为Entity和Virtual，如果取值为Virtual时，默认的持久化不作保存；默认值为Virtual。
- SecondaryType 数据对象的辅助类型，在PrimaryType为Virtual时无意义；用于说明数据对象的用处，取值如下：
  - Normal 普通实体数据；
  - Dict 字典数据；
  - Migration 迁移表；
  - CompDict 复合字典；
  - ChainDict 链式字典。

  默认值为Normal。
- PrimaryTableKey 主表标识，在PrimaryType为Virtual时无效；
- NoPrefix 数据对象的编号前缀；

13.3:OID过滤条件

```
<OIDFilter Type="" Formula="" Impl="">
    <Statement>
        <![CDATA[ ]]
    </Statement>
    <ParameterCollection>
    </ParameterCollection>
</OIDFilter>
```

OIDFilter定义实体表单如何在中间层确定OID的值。其包括以下属性：

- Type 过滤的类型，取值为：
  - Query 来源于过滤，即OID的值通过查询语句的执行结果来确定，OID的取值为结果集的第0行第0列的值。
  - Formula 来源于表达式，OID的值为执行结果的返回值；
  - Interface 来源于Java类的执行结果，暂时未定义接口类。
- Formula Type为Formula时定义表达式的内容；
- Impl Type为Interface时定义实现类的全路径名；
- Statement 节点为Type等于Query时查询语句的内容；
- ParameterCollection 为参数集，具体见表中的定义。

13.4:表集合定义

```
<TableCollection>
```

```
    <Table />
    ...
</TableCollection>
```

表集合由多个表组成。

13.4.1:表定义

```
<Table>
    <Column />
    ...
</Table>
```

每个表由多个列组成。表的属性如下：

```
<Table Key="" Caption="" DBTableName="" TableMode="" SourceType="" Persist="" Formula="" Impl="">
    <Statement>
        <![CDATA[ ... ]]!>
    </Statement>
    <TableFilter>
        <![CDATA[ ]]
    </TableFilter>
    <ParameterCollection>
    </ParameterCollection>
    <TableSourceCollection>
    </TableSourceCollection>
</Table>
```

- Key 表的标识，在当前数据对象范围内唯一；
- Caption 表的名称，对表的说明；
- DBTableName 对应的数据库表的名称，在未定义的情况下，对应的数据库表的名称为Key的值；
- TableMode 表的模式，取值如下：
  - Head 头表，表的数据行最多只有一行；
  - Detail 明细表，表的数据行为多行；

  在数据对象为虚拟表时取值无意义；
- SourceType 数据的来源类型，取值如下：
  - Table 来源于表；
  - Query 来源于查询语句；
  - Custom 来源于表达式执行的结果；（未实现，扩充用）
  - Interface 来源于二次开发类执行的结果；（未实现，扩充用）

  默认值为Table；
- Persist 是否持久化，取值为True或False，默认值为True；
- TableFilter 定义默认的过滤条件；
- ParentKey 父表标识，在表之间存在父子关系时定义；
- Hidden 隐藏属性，取值为True和False，为True时表示只作为定义使用，不作为保存和查询的的结果列；默认值为False。
- Formula SourceType类型为Formula时定义表达式的内容；
- Impl SourceType为Interface时定义接口类的名称，必须是com.bokesoft.yigo.mid.extend.IMidProcess的实现类。

- Statement 在这里作为一个节点而存在，在其CDATA片段中定义SQL语句。
- ParameterCollection 定义默认的查询参数集，为SQL语句中的?参数定义；后面详细介绍。
- TableSourceCollection 表的数据来源集合。

13.4.2:参数集

```
<ParameterCollection>
    <Parameter TargetTable="" TargetColumn=""  DataType="" SourceType="" Formula="" FieldKey="" Value="" Description=""/>
        ...
</ParameterCollection>
```

参数集包含多个参数定义，参数的属性如下：

- TargetTable 参数对应的目标表；(未实现，扩充用)
- TargetColumn 参数的对应的目标列，可以不定义，不定义的情况下，根据DataType定义来确定参数的类型，如果在DataType也未定义的情况下，根据实际值来确定参数类型；
- DataType 定义参数的类型，在定义了TargetColumn的情况下，通过列的定义推算；可取值为列的类型定义中的所有类型，见列定义。
- SourceType 参数的来源类型，取值如下：
  - Formula 来源于表达式；
  - Field 来源于字段；
  - Const 来源于常量；

  默认值为Formula。
- Formula 来源于表达式的情况下定义表达式内容；
- FieldKey 来源于字段的情况下定义字段(组件标识、ListView的列或Grid中的单元格)；
- Value 来源于常量时定义常量值；
- Description 对参数的描述。

13.4.3:表来源集合

```
<TableSourceCollection>
    <TableSource Key="" Description="" SourceType="" Rule="" Formula="" Impl="" LinkTable="">
        <TableFilter>
            <![CDATA[ ]]
        </TableFilter>
        <ParameterCollection>
        </ParameterCollection>
    </RuleFilter>
    ...
</TableSourceCollection>
```

规则过滤集由多个按规则匹配的规则组成，当存在一个满足条件的过滤时，则表表使用此过滤进行数据查询，否则使用默认的过滤条件。每个规则过滤通过RuleFilter定义。其属性如下：

- Key 来源的标识，通过这个标识可以在载入数据的时候指定来源；
- Description 描述；
- SourceType 同表中的同名定义，多一个LinkTable(来源于关联表)定义；
- Rule 规则，通过此规则决定是否使用此数据来源；

- Formula 同表中的同名定义；
- Impl 同表中的同名定义；
- LinkTable 关联表，在SourceType为LinkTable时指定关联表的标识；
- TableFilter 为过滤的具体内容；
- ParameterCollection 为过滤的参数集，具体见参数集说明。

13.4.4:列定义

```
<Column Key="" Caption="" Description="" Persist="" DBColumnName="" DataType="" Length="" Precision="" Scale="" IsPrimary=""
 Cache=""
    DefaultValue="" DefaultFormulaValue="" RightsType="" Hidden="" IsGroup="" GroupType="" SplitType="">
</Column>
```

- Key 列的标识，表内不能重复；
- Caption 列的名称，列的简要性描述；
- Description 列的描述，详细的说明；
- Persist 是否持久化，取值为True和False，默认值为True；
- DBColumnName 对应的数据库列的名称，如果未定义，对应的数据列的名称取Key的值；
- DataType 数据的类型，取值范围如下：
  - Long 长整型；
  - Integer 整型；
  - Varchar 变长字符串；
  - Numeric 数值类型；
  - DateTime 日期类型；
  - Binary 二进制类型；
- Length 在DataType为Varchar和Binary时定义长度；
- Precision 在数据类型为Numeric时定义精度；
- Scale 在数据类型为Numeric时定义小数位数；
- IsPrimary 业务关键字标志，取值为True和False，默认值为False；
- Cache 缓存标志，取值为True和False，如果取值为True则在字典缓存中存储该列的值；默认值为False；
- DefaultValue 列的默认值；
- DefaultFormulaValue 表达式形式定义的默认值；
- NeedRights 定义是否需要处理权限，取值为True和False，默认值为False；
- ItemKey 权限数据关联的字典标识；
- RightsType 权限的处理类型，取值为Remove和Deny；默认值为空；
- Hidden 隐藏标志，取值为True和False，定义为True是仅作为定义使用，在数据载入及存储时均不作处理；默认值为False；

- IsGroup 是否分组字段，在作为迁移表时定义，取值为True和False，默认值为False；
- GroupType 分组类型，为分组字段时定义，取值为Discrete(直接量)和Period(期间)；默认值为空；
- SplitType 字段的拆分类型，取值为Period，设置该属性列会被额外增加期初和期末两列，只对迁移表有效；默认值为空；
- AccessControl 访问控制标志，取值为True和False，定义为True时，该字段会生成Key+_CP的列来存储访问控制位；
- Sort 排序类型，取值如下：
  - None 无排序；
  - Asc 升序；
  - Desc 降序；

13.4.5:系统列说明

对于数据对象中每个需要存储的表，必须定义如下系统字段：

- SOID Long类型，系统OID；
- OID Long类型，对象OID；
- POID Long类型，父对象OID；
- VERID Integer类型，数据版本号；
- DVERID Integer类型，明细控制版本号；

对于需要存储的列，以下列定义具有确定的意义：

- CREATETIME - 创建时间；
- MODIFYTIME - 修改时间；
- HAPPENTIME - 发生时间；
- CREATOR - 创建人；
- MODIFIER - 修改人；
- NO - 编号字段；
- LAYER - 数据层次标识；
- HIDDEN - 隐藏标志标识；同LAYER字段配合使用，该字段不存储，在定义时不要配置为Persist为True。

13.5:关系定义

关系定义复合字典的数据来源，关系数据对象的标识用于复合字典；

```
<Relation Caption="" Description="">
    <Layer />
    ...
</Relation>
```

关系由多个层次组成，关系自身的属性如下：

- Caption 关系的名称；
- Description 关系的详细描述；

13.5.1:层次定义

```
<Layer ItemKey="" Relation="" TableKey="" ColumnKey=""/>
```

- ItemKey 字典的标识，这里不可以再引用复合字典；
- Relation 关系的类型，取值如下：
  - Contain 包含，即上层节点属性在包含下层节点的列表；
  - Belong 从属，下层节点定义属性哪个上层节点。
- TableKey 表标识，即定义包含或从属关系的字段所在的表；
- ColumnKey 列标识，定义包含或从属关系的字段所在的列；

13.6:检查规则集合

```
<CheckRuleCollection>
    <CheckRule ErrorInfo="">
        <![CDATA[ ... ]]!>
    </CheckRule>
</CheckRuleCollection>
```

每个数据对象包含多个检查规则，规则的属性如下：

- ErrorInfo 检查规则的错误描述信息；
- CDATA部分的检查规则内容，内容为表达式。

13.7:处理

```
<!-- 这里ProcessTag不是实际的标签名，只是代表实现的处理的标签 -->
<ProcessTag>
    <Process>
        <![CDATA[ ... ]]!>
    </Process>
</ProcessTag>
```

分别包括PreLoadProcess、PostLoadProcess、PreSaveProcess、PostSaveProcess，除了标签名不同外，其属性均相同，属性如下：

- Description 处理的描述；
- CDATA部分的处理的定义内容，表达式；

13.8:二次开发扩展

```
<ExtendCollection>
    <Extend Class="" Alias=""/>
    ...
</ExtendCollection>
```

每个数据对象可以定义多个二次开发扩展类；Extend属性如下：

- Class 类的全路径类，类必须继承com.bokesoft.yes.mid.dev.MidExtend类；
- Alias 类的别名，在表达式中用于访问类的函数；

第 14 章 数据迁移属性

目录

14.1:结构

```
<DataMigration>
    <!-- 源表集合 -->
    <SourceTableCollection>
    </SourceTableCollection>
    <!-- 目标表集合 -->
    <TargetTableCollection>
    </TargetTableCollection>
</DataMigration>
```

14.2:迁移属性

```
<DataMigration Key="" Caption="" Description="" SrcDataObjectKey="" TgtDataObjectKey="" StatusFieldKey="" StatusValue=""
 Condition="">
</DataMigration>
```

- Key 迁移标识，全局唯一；
- Caption 迁移名称；
- Description 迁移详细说明；
- SrcDataObjectKey 源数据对象标识；
- TgtDataObjectKey 目标数据对象标识；
- StatusFieldKey 状态字段标识；
- StatusValue 迁移的条件阀值；
- Condition 迁移的数据条件；

14.3:源表集合

```
<SourceTableCollection X="" Y="" Width="" Height="">
    <SourceTable TableKey="" IsPrimary="">
        <SourceField Type="" Definition="" OpSign="" IsNegtive="" GroupingPolicy=""
               PeriodGranularity="" PeriodValue="" MapFormula="" TargetTableKey="" TargetFieldKey="" RefFieldKey=""/>
        ...
    </SourceTable>
    ...
</SourceTableCollection>
```

表集合属性：

- X 在设计界面上的水平位置；
- Y 在设计界面上的垂直位置；
- Width 在设计界面上的宽度；

- Height 在设计界面上的高度；

源表集合包含多个源表定义，源表属性如下：

- TableKey 表标识；
- IsPrimary 是否迁移的主表；

每个源表包含多个需要迁移的字段，属性如下：

- Type 字段类型，取值如下：
  - Field 字段，即值来源于表中列的值；
  - Formula 表达式，值来源于表达式的计算结果；
  - Const 常量，值来源于常量定义；

  默认值为Field；
- Definition 字段的定义，Type取值为Field时定义表中列的标识；Type取值为Formula时定义表达式的内容；Type为Const时定义常量的值；
- OpSign 值处理方式，取值如下：
  - AddDelta 加变化量；
  - AddValue 加直接量；
  - Assign 赋直接量；

  默认值为AddDelta；该定义对分组字段无效；
- IsNegtive 是否负向迁移，取值为True和False，默认值为False；
- GroupingPolicy 分组策略，取值如下：
  - Discrete 直接量；
  - Peroid 期间分组；

  默认值为Discrete；
- PeriodGranularity 期间粒度，取值如下：
  - Year 自然年；
  - Month 自然月；
  - Day 自然日；
  - FiscalMonth 会计月份；
- PeriodValue 期间取值方式，取值如下：
  - Default 默认，即取字段值根据期间粒度进行换算；
  - Value 直接取值，即字段值就是期间值；
  - Map 值映射；即通过值映射公式换算期间值；
- MapFormula 值映射公式；
- TargetTableKey 目标表标识；

- TargetFieldKey 目标字段标识；
- RefFieldKey 参数的主表字段标识，SourceTable中所有具有该属性的字段形成跟主表之间的数据对应关系；

14.4:目标表集合

```
<TargetTableCollection X="" Y="" Width="" Height="">
    <TargetTable TableKey="">
        <TargetField FieldKey=""/>
        ...
    </TargetTable>
    ...
</TargetTableCollection>
```

目标表集合属性：

- X 在设计器界面上的水平位置；
- Y 在设计器界面上的垂直位置；
- Width 在设计器界面上的宽度；
- Height 在设计器界面上的高度；

目标表集合由多个表组成，表的属性如下：

- TableKey 表的标识；

目标表由多个字段组成，字段的属性如下：

- FieldKey 列的标识；

第 15 章 数据映射属性

目录

15.1:结构

```
<Map>
    <!-- 源表集合 -->
    <SourceTableCollection>
    </SourceTableCollection>
    <!-- 目标表集合 -->
    <TargetTableCollection>
    </TargetTableCollection>
    <!-- 反填集合 -->
    <FeedbackCollection>
    </FeedbackCollection>
</Map>
```

说明如下：

- SourceTableCollection 为源表集合，定义映射的来源表；
- TargetTableCollection 为目标表集合，定义映射的目标表；
- FeedbackCollection 为反填定义集合；

15.2:映射属性

```
<Map Key="" Caption="" SrcDataObjectKey="" TgtDataObjectKey="" MaxPushValue="" MinPushValue="" AllowSurplus="" SurplusLimit=""
 AllowSurplusSave=""/>
```

- Key 标识；
- Caption 名称；
- SrcDataObjectKey 源数据对象标识；
- TgtDataObjectKey 目标数据对象标识；
- MaxPushValue 最大下推值，表达式；
- MinPushValue 最小下推值，表达式；
- AllowSurplus 是否允许超量，取值为True和False，默认值为False；
- SurplusLimit 超量上限，表达式；
- AllowSurplusSave 是否允许超量保存，取值为True和False，默认值为False；

15.3:源表集合

```
<SourceTableCollection X="" Y="" Width="" Height="">
    <SourceTable Key="" IsPrimary="" TargetTableKey="">
```

```
        <SourceField Type="" Definition="" EdgeType="" Condition="" Editable=""
            TargetTableKey="" TargetFieldKey="" RefFieldKey=""/>
            ...
    </SourceTable>
    ...
</SourceTableCollection>
```

表集合属性

- X 在设计界面上的水平位置；
- Y 在设计界面上的垂直位置；
- Width 在设计界面上的宽度；
- Height 在设计界面上的高度；

表属性

- Key 表标识；
- IsPrimary 是否主表标志，取值为True和False，默认值为False；
- TargetTableKey 目标表标识；

源字段属性

- Type 字段类型，取值如下：
  - Field 字段；
  - Formula 公式；
  - Const 常量；
- EdgeType 映射边的类型，取值如下：
  - Normal 普通数据映射；
  - Focus 关注字段映射；
  - Relation 关系映射；
  - MapKey 关系标识映射；
- Condition 字段的映射条件，默认值为True；
- Editable 下推目标是否可编辑，默认值为True；
- TargetTableKey 目标表标识；
- TargetFieldKey 目标字段标识；
- RefFieldKey - 参照的主表字段标识，SourceTable中所有具有该属性的字段形成跟主表之间的数据对应关系；

15.4:目标表集合

```
<TargetTableCollection X="" Y="" Width="" Height="">
    <TargetTable Key="">
        <TargetField Type="" Definition="" Lock="">
            <Feedback OpSign="" ObjectKey="" TableKey="" FieldKey="">
                <PostTrigger>
                    <![CDATA[com.bokesoft.XXX]]>
                <PostTrigger/>
```

```
            <Feedback/>
            ...
        </TargetField>
        ...
    </TargetTable>
    ...
</TargetTableCollection>
```

表集合属性

- X 在设计界面上的水平位置；
- Y 在设计界面上的垂直位置；
- Width 在设计界面上的宽度；
- Height 在设计界面上的高度；

表属性

- Key 表标识；

字段属性

- Type 字段的类型，取值如下：
  - Field 字段；
  - Formula 公式；
  - Const 常量；
- Definition Type取值为Field时为字段的标识，取值为Formula时为表达式的内容，取值为Const时为常量的字面量；
- Lock 字段是否锁定，取值为True和False，默认值为False。

反填属性

- ObjectKey 反填目标数据对象标识；
- TableKey 反填的目标表标识；
- FieldKey 反填的目标字段标识；
- OpSign 反填的数据处理方式，取值如下：
  - AddDelta 加变化量；
  - Assign 直接赋值；
- PostTrigger 反填的后期处理类名称。

15.5:反填集合

```
<FeedbackCollection StatusFieldKey="" StatusFieldValue="" Condition="">
    <FeedbackObject ObjectKey="" X="" Y="" Width="" Height="">
        <FeedbackTable TableKey="TableKey">
            <FeedbackField FieldKey=""/>
        </FeedbackTable>
        ...
    </FeedbackObject>
    ...
</FeedbackCollection>
```

反填属性

- StatusFieldKey 状态字段标识；
- StatusValue 状态条件值；
- Condition 数据条件；

反填目标对象属性(设计属性)

- ObjectKey 数据对象标识；
- X 在设计界面上的水平位置；
- Y 在设计界面上的垂直位置；
- Width 在设计界面上的宽度；
- Height 在设计界面上的高度；

反填表属性(设计属性)

- TableKey 表标识；

反填字段属性(设计属性)

- FieldKey 字段标识；

第 7 部分 界面属性参考

目录

第 16 章 表单属性

目录

16.1:表单定义结构

表单定义结构如下所示：

```
<Form>
    <!-- 数据源定义 -->
    <DataSource>
    </DataSource>
    <!-- 表单加载事件 -->
    <OnLoad>
        <![CDATA[...]]>
    </OnLoad>
    <!-- 脚本定义 -->
    <ScriptCollection>
    </ScriptCollection>
    <!-- 操作定义 -->
    <OperationCollection>
    </OperationCollection>
    <!-- 二次开发扩展定义 -->
    <ExtendCollection>
    </ExtendCollection>
    <!-- 宏公式定义 -->
    <MacroCollection>
    </MacroCollection>
    <!-- 状态集合定义 -->
    <StatusCollection>
    </StatusCollection>
    <!-- 界面检查规则集合 -->
    <UICheckRuleCollection>
    </UICheckRuleCollection>
    <!-- 界面主体定义 -->
    <Body>
        <!-- 布局集合 -->
        <LayoutCollection>
        </LayoutCollection>
        <!-- 表单构造 -->
        <Block>
            <!-- 根面板 -->
            <Component />
        </Block>
    </Body>
</Form>
```

说明：

- DataSource 为表单的数据源，可以在同一个文件内部定义数据源，也可以引用外部数据源；
- OnLoad 表单的加载事件；
- ScriptCollection 脚本定义，这里主要定义具有一些确定意义的脚本定义，比如表单的载入和保存脚本；
- OperationCollection 操作定义，定义在默认的工具栏中需要加载的用户操作按钮；
- ExtendCollection 界面二次开发需要引用的类的集合；

- MacroCollection 公共的宏公式定义；
- StatusCollection 表单自有状态集合定义；
- UICheckRuleCollection 界面检查规则集合；
- Body 为表单的界面定义；
- LayoutCollection 为表单的布局集合；
- Block 为表单构造块，设计上允许有多个构造块，目前仅支持一个。
- Component 构造块中的根组件，只能是面板。

16.2:表单属性

```
<Form Key="" Caption="" FormulaCaption="" AbbrCaption="" FormulaAbbrCaption="" FormType="" Platform="" Shell=""/>
```

- Key 表单标识；
- Caption 表单名称；
- FormulaCaption 以表达式表示的表单名称；
- AbbrCaption 表单的缩写名称；
- FormulaAbbrCaption 以表达式表示的表单缩写名称；
- FormType 表单的类型，取值如下：
  - Normal 普通表单；
  - Entity 实体表单；
  - View 视图表单；
  - Condition 查询条件表单；
  - Detail 明细表单；
- Platform 支持的平台列表；
- Shell 表单的壳提供程序定义，由各个平台单独实现，用于二次开发；

16.3:DataSource数据源属性

```
<Form>
    <DataSource RefObjectKey="RefObjectKey" RefTableKey="">
        [内嵌对象定义
        <DataObject>
        ...
        </DataObject>
        ]
    </DataSource>
    ...
</Form>
```

- RefObjectKey 引用的数据对象标识；
- RefTableKey 引用的数据表标识；

注意

如果RefObjectKey为空，那么表单中可能定义内嵌的数据对象，此时数据对象定义被包含在DataSource节点下；

16.4:ScriptCollection脚本集合

同应用及工程定义中的脚本定义集合。

16.5:UICheckRuleCollection(界面全局检查规则集合)

```
<UICheckCollection>
    <UICheckRule Description="" ErrorInfo="">
        <![CDATA[...]]
    </UICheckRule>
    ...
</UICheckCollection>
```

全局检查规则集合包含多个界面检查规则，界面检查规则的属性定义如下：

- Description 规则的描述；
- ErrorInfo 错误描述；
- CDATA为规则的具体内容；

16.6:ExtendCollection及Extend

```
<ExtendCollection>
    <Extend Class="" Alias=""/>
    ...
</ExtendCollection>
```

界面二次开发扩展集合包含多个扩展定义，扩展定义属性如下：

- Class 二次开发扩展的全路径名，为继承com.bokesoft.yes.view.dev.FormExtend类的子类；
- Alias 类的别名，方便在表达式中引用；

16.7:表单主体属性(Body)

```
<Body Width="" Height="" PopWidth="" PopHeight="" HAlign="" OverflowX="" OverflowY="" TopMargin="" BottomMargin=""
 LeftMargin="" RightMargin="">
    ...
</Body>
```

- Width - 宽度，见尺寸定义，取值可以为比例、固定值；
- Height - 高度，见尺寸定义，取值可以为比例、固定值。
- PopWidth - 弹出窗口宽度，取值为固定值；
- PopHeight - 弹出窗口高度，取值为固定值；
- HAlign - 对齐方式，取值为Left,Center和Right。
- OverflowX - 水平超出时的显示属性，取值为Visible，Scroll，Hidden，Auto；
- OverflowY - 垂直超出时的显示属性，取值为Visible，Scroll，Hidden，Auto；
- TopMargin 顶边距；
- BottomMargin 底边距；
- LeftMargin 左边距；

- RightMargin 右边距;

第 17 章 表单组件属性

目录

17.1:组件公共属性

```
<Component>
    <DataBinding>
    </DataBinding>
    <Activate>
        <![CDATA[ ... ]]
    </Activate>
    <Format>
    </Format>
</Comonent>
```

17.1.1:基本属性

```
<Component Key="" Caption="" BuddyKey="" Visible="" Enable="" X="" Y="" XSpan="" YSpan="" Width="" Height=""
           HAlign="" VAlign="" Class="" Position="" Area=""  Tip=""
           Padding="" LeftPadding="" TopPadding="" RightPadding="" BottomPadding=""
           Margin="" LeftMargin="" TopMargin="" RightMargin="" BottomMargin="" HasBorder="" TabOrder="">
</Component>
```

- Key 组件的标识;
- Caption 组件的名称;
- BuddyKey 伙伴组件的标识;
- Visible 可见性，无默认值;
- Enable 可用性，无默认值;
- X 水平位置(在按网格位置布局的面板中使用);
- Y 垂直位置(在按网格位置布局的面板中使用);
- XSpan 水平跨度(在按网格位置布局的面板中使用);
- YSpan 垂直跨度(在按网格位置布局的面板中使用);
- Width 组件的宽度;
- Height 组件的高度;
- HAlign 水平对齐方式，不是文本的对齐方式；取值为Left、Center和Right，默认值为Center;
- VAlign 垂直对齐方式，不是文本的对齐方式；取值为Top、Center和Bottom，默认值为Top;
- Class 样式类的名称;
- Position - 组件的定位属性，取值为同HTML DOM position属性，这里列出Static、Relative、Absolute、Fixed;
- Tip - 提示文本;

- Padding - 填充，定义顶、右、底、左四个方位的填充；
- LeftPadding - 左方位填充，优先级高于Padding；
- TopPadding - 顶方位填充，优先级高于Padding；
- RightPadding - 右方位填充，优先级高于Padding；
- BottomPadding - 底方位填充，优先级高于Padding；
- Magin - 边距，定义顶、右、底、左四个方位的边距(仅portal中使用)；
- LeftMargin - 左边距，优先级高于Margin(仅portal中使用)；
- TopMargin - 顶边距，优先级高于Margin(仅portal中使用)；
- RightMargin - 右边距，优先级高于Margin(仅portal中使用)；
- BottomMargin - 底边距，优先级高于Margin(仅portal中使用)；
- HasBorder - 是否有边框，默认为True；
- BorderColor - 边框颜色，默认为空；
- BorderRadius - 边框圆角；
- TabOrder - 组件的焦点顺序，默认值为-1；

17.1.2:DataBinding属性

```
<DataBinding Required="" TableKey="TableKey" ColumnKey="ColumnKey" ValueChanged="" DefaultValue=""
           DefaultFormulaValue="" ValueDependency="" CheckRule="" CheckDependency="" ErrorInfo="">
</DataBinding>
```

- TableKey 绑定表的标识；
- ColumnKey 绑定列的标识；
- Required 是否必填项，默认值为False；
- ValueChanged - 值改变事件；
- DefaultValue - 默认值；
- DefaultFormulaValue - 表达式默认值；
- ValueDependency - 值关联域集合；(值表达式中依赖关系无法解析时使用)
- CheckRule - 检查规则；
- CheckRuleDependency - 检查规则关联域集合；(依赖关系无法解析时使用)
- ErrorInfo - 错误描述；

17.1.3:Activate事件

在组件活动时触发，在组件作为选项卡的子项时，如果子项选中，激发该事件。

17.1.4:Format格式

```
<Format ForeColor="ForeColor" BackColor="BackColor">
    <Font Name="Name" Size="Size" Bold="Bold" Italic="Italic"/>
</Format>
```

格式：

- ForeColor - 前景色；
- BackColor - 背景色；

字体：

- Name - 字体名称；
- Size - 字体大小；
- Bold - 是否粗体；
- Italic - 是否斜体。

17.2:面板公共属性

```
<Panel  OverflowX="" OverflowY="" BackImage="" BackImagePosition="" BackImageRepeat="">
</Panel>
```

除具有组件的属性外，面板还具有以下属性：

- OverflowX - 水平超出时的显示属性，取值为Visible，Scroll，Hidden，Auto；
- OverflowY - 垂直超出时的显示属性，取值为Visible，Scroll，Hidden，Auto；
- BackImage - 背景图片；
- BackImagePosition - 背景图片位置，取值为Left,Top,Right,Botom,Center；默认为Center；
- BackImageRepeat - 背景图片是否垂直填充；取值为True和False，默认为False；

17.3:面板

17.3.1:列式布局面板(ColumnLayoutPanel)

```
<ColumnLayoutPanel>
    <Component X="" Y="" XSpan=""/>
    ...
</ColumnLayoutPanel>
```

列式布局面板通过<ColumnLayoutPanel>标记来定义，列式布局面板将区域划分成多个12列均分的行，通过其包含的子组件的如下属性来确定的位置：

- X 确定在行中的水平位置；
- Y 确定行号；
- XSpan 确定水平列跨度；

17.3.2:边界布局面板(BorderLayoutPanel)

```
<BorderLayoutPanel>
    <Component Area=""/>
    ...
</BorderLayoutPanel>
```

边界布局面板通过<BorderLayoutPanel>标记来定义，其通过子组件的如下属性来确定位置：

- Area 组件的位置，取值为Left、Right、Top、Bottom和Center。

17.3.3:流布局面板(FlowLayoutPanel)

```
<FlowLayoutPanel>
    <Component />
    ...
</FlowLayoutPanel>
```

流布局面板通过<FlowLayoutPanel>标记来定义，其通过子组件定义的顺序从下到下排列子组件。

子组件的高度计算如下：如果定义高度为px，那么取其定义值；如果定义高度为比例，则取面板的高度乘以该比例；如果定义为pref或未定义，取其内部高度。

17.3.4:选项卡面板(TabPanel)

```
<TabPanel>
    <Component/>
    ...
</TabPanel>
```

选项卡面板通过<TabPanel>标记定义，其通过子组件的定义顺序从左到右按选项卡组件组件。

17.3.5:拆分布局(SplitPanel)

```
<SplitPanel Orientation="Orientation">
    <SplitSize Size="Size"/>
    ...
    <Component />
    <Component />
    ...
</SplitPanel>
```

面板属性：

- Orientation 表示拆分方向，取值为Horizontal或Vertical。默认值为Horizontal。

拆分定义(SplitSize)属性：

- Size - 表示子组件的大小，取值为固定值或百分比，比如固定值200px，比例值100%；

17.3.6:网格布局面板(GridLayoutPanel)

```
<GridLayoutPanel ForceLayout="">
    <RowDefCollection RowHeight="" RowGap="">
        <RowDef Height="" />
        ...
    </RowDefCollection>
    <ColumnDefCollection>
        <ColumnDef Width="" />
        ...
    </ColumnDefCollection>
    <Component X="" Y="" XSpan="" YSpan=""/>
    ...
</GridLayoutPanel>
```

网格布局面板将整个区域划分成多行多列的网格，子组件在网格内根据位置和跨度分布，面板的属性如下：

- ForceLayout 用于定义网格布局是否使用自有的布局策略，如果为True表示使用自己的布局策略，否则使用客户端环境的布局策略(比如使用浏览器的布局)。默认值为True；

行定义集合(RowDefCollection)属性如下：

- RowHeight默认行高；
- RowGap默认行间距；

行属性(RowDef)如下：

- Height　　行高，取值为固定值、比例、继承(取RowDefCollection中的RowHeight值)或自适应，比如：固定值120px，比例值50%，继承inherit，自适应内容pref。默认值为inherit。

列定义集合(ColumnDefCollection)包含多个列定义，列定义属性如下：

- Width 列宽，取值为固定(px)、比例(%)、自动分配(auto)，默认值为自动分配(auto)。其中在网格中自动分配的列和比例的列不能混合使用。

子组件中跟布局相关的属性如下：

- X 网格中的水平位置；
- Y 网格中的垂直位置；
- XSpan 网格中的水平跨度；
- YSpan 网格上的垂直跨度；

17.3.7:线性布局面板(LinearLayoutPanel)

```
<LinearLayoutPanel Key="" Orientation="">
    <Component />
    ...
</LinearLayoutPanel>
```

线性布局为Android设备专用布局，具有属性：

- Orientation 线性布局的方向，取值为VERTICAL和HORIZONTAL，默认值为HORIZONAL。

线性布局中的组件位置根据组件的宽度或高度来进行，宽度或高度为固定值或比例值；以横向为例，组件的宽度为固定值时，则其高度直接取固定值，如果为比例值，则组件的宽度为面板的宽度乘以该比例。

17.3.8:流式表布局面板(FluidTableLayoutPanel)

```
<FluidTableLayoutPanel RepeatCount="" RepeatGap="" RowGap="" ColumnGap="" RowHeight="">
    <TableColumnCollection>
        <TableColumn Width="def" />
        ...
    </TableColumnCollection>
    <Component />
    ...
</FluidTableLayoutPanel>
```

属性如下：

- RepeatCount 为列向的重复数，用于定义布局在列向的重复次数；
- RepeatGap 为列向的间距；
- RowGap 为行间距；
- ColumnGap 为列间距；
- RowHeight 为行高定义，尺寸；

列定义集合(TableColumnCollection)由多个列定义组成，属性如下：

- Width 为布局列的宽度，固定值或比例。

17.3.9:弹性流布局面板(FlexFlowLayoutPanel)

```
<FlexFlowLayoutPanel>
```

```
    <Component Height=""/>
    ...
</FlexFlowLayoutPanel>
```

弹性流布局中的组件位置根据组件的高度(Height)来进行，高度为固定值或比例值，比例值的计算是面板的高度减去所有的固定值或者自适应值，余下的高度乘以相应的比例值。

17.4:普通控件

17.4.1:静态文本(Label)

```
<Label Icon="Icon">
</Label>
```

- Icon 静态文本的图标。

17.4.2:分隔线(Separator)

```
<Separator>
</Separator>
```

17.4.3:按钮(Button)

```
<Button Icon="Icon" PrimayIcon="" SecondaryIcon="">
    <OnClick>
        <![CDATA[ ]]
    </OnClick>
</Button>
```

- Icon 按钮图标;
- PrimayIcon 按钮左边图标;
- SecondaryIcon 按钮右边图标;
- OnClick 按钮点击事件；脚本类型。

17.4.4:拆分按钮(SplitButton)

```
<SplitButton Icon="Icon">
    <OnClick>
        <![CDATA[ ]]>
    </OnClick>
    <DropdownItem Text="" Separator="">
        <OnClick>
            <![CDATA[ ]]>
        </OnClick>
    </DropdownItem>
    ...
</SplitButton>
```

按钮属性

- Icon 按钮的图标;
- OnClick 按钮的点击事件，脚本类型;

下拉项(DropdownItem)属性

- Text 下拉项的文本;
- 是否分隔项，取值为True或False，默认为False;
- OnClick 下拉项的点击事件，脚本类型;

17.4.5:下拉按钮(DropdownButton)

```
<DropdownButton Icon="Icon">
    <DropdownItem Text="" Separator="">
        <OnClick>
            <![CDATA[  ]]>
        </OnClick>
    </DropdownItem>
    ...
</DropdownButton>
```

按钮属性

- Icon 下拉按钮的图标。

下拉项(DropdownItem)属性

- Text 下拉项的文本;
- 是否分隔项，取值为True或False，默认为False;
- OnClick 下拉项的点击事件，脚本类型;

17.4.6:文本编辑框(TextEditor)

```
<TextEditor MaxLength="" InvalidChars="" Case="" PromptText="" Trim="" SelectOnFocus="" Mask="" Icon="" PreIcon=""
 EmbedText="" HoldFocus="">
    <KeyEnter>
        <![CDATA[  ]]>
    </KeyEnter>
</TextEditor>
```

- MaxLength - 可输入文本的最大长度;
- InvalidChars - 不允许办理入的字符列表;
- Case - 是否大小写转换，取值为None,Lower,Upper,默认值为None;
- PromptText - 内容为空的时候的提示文本;
- Trim - 是否去除首尾多余空格;
- SelectOnFocus - 光标进入默认全选，取值为True和False，默认值为True;
- Icon - 右侧图标url;
- PreIcon - 左侧图标url;
- EmbedText - 左侧内嵌文本;
- HoldFocus 保持焦点属性，取值为True和False，默认值为False，只对头控件有效;
- KeyEnter 节点定义回车键事件，只对头控件有效;

17.4.7:文本域(TextArea)

```
<TextArea MaxLength="">
</TextArea>
```

- MaxLength 允许输入的文本长度。

17.4.8:密码编辑框(PasswordEditor)

```
<PasswordEditor PreIcon="" EmbedText="">
```

```
</PasswordEditor>
```

- PreIcon 左侧图标路径；
- EmbedText 内嵌文本。

17.4.9:数值编辑框(NumberEditor)

```
<NumberEditor Precision="" Scale="" ScaleSeparator="" UseGroupingSeparator="" GroupingSize="" GroupingSeparator=""
        PromptText="" SelectOnFocus="" RoundingMode="" SelectOnFocus="" ZeroString="" NegtiveForeColor="">
</NumberEditor>
```

- Precision - 精度，默认值16；
- Scale - 小数位位数，默认值2；
- UseGroupingSeparator - 是否使用组分隔，比如千分位，取值为True和False，默认值为True；
- GroupingSize - 组大小，默认值为3；
- GroupingSeparator - 组分隔符，默认值为逗号(,)；
- ScaleSeparator - 小数位分隔符号，默认值为点(.)；
- PromptText - 内容为空的时候的提示文本；
- RoundingMode 舍入规则，取值如下：
  - HALF_UP 向最接近整数的进位，相当于四舍五入；
  - ROUND_UP 总是向最大整数方向进位；
  - ROUND_DOWN 总是向最小整数方向进位；
- SelectOnFocus - 在获得焦点时是否全选内容，默认值为True；
- ZeroString 0值显示字符串；
- NegtiveForeColor 负数前景色。

17.4.10:复选框(CheckBox)

```
<CheckBox Checked="">
</CheckBox>
```

- Checked - 是否默认选中，取值为True和False，默认值是False。

17.4.11:单选框(RadioButton)

```
<RadioButton GroupKey="" Checked="" IsGroupHead="" Value="">
</RadioButton>
```

- GroupKey 组标识；
- Checked 是否默认选中，取值为True和False，默认值为False；
- IsGroupHead 单选框组头标志，取值为True，False；默认值为False；
- Value 选中的值。

17.4.12:文本按钮(TextButton)

```
<TextButton MaxLength="" InvalidChars="" Case="">
```

```
    <OnClick>
        <![CDATA[  ]]
    </OnClick>
</TextButton>
```

- MaxLength 可输入文本的最大长度；
- InvalidChars 不允许输入的字符列表；
- Case 是否大小写转换，取值为None,Lower,Upper,默认值为None；
- PromptText 内容为空时的提示文本；
- OnClick 按钮点击事件。

17.4.13:字典(Dict)

```
<Dict ItemKey="" AllowMultiSelection="" Root="" Independent="" DisSelect="" AutoLocate="">
    <ItemFilter ItemKey="">
        <Filter Type="" Query="" Op="" Condition=""  FilterDependency="">
            <FilterValue FieldKey="" Type="" RefValue="" ParaValue=""/>
            ...
        </Filter>
        ...
    </ItemFilter>
    ...
</Dict>
```

字典属性

- ItemKey 字典对象的标识；
- Root - 字典树显示的根节点，可以指定一个汇总节点作为字典树根节点，取值来源为字典组件；
- AllowMultiSelection - 是否允许多选；
- Independent - 父子节点是否联动，true的时候不联动， false的时候联动 默认为false；
- AutoLocate - 自动定位，默认值为False；

字典过滤(ItemFilter)

- ItemKey 字典项标识，只对复合字典有效；

过滤项属性(Filter)

- Type 过滤项的类型，取值如下：
  - Field 字段值关联；
  - DataSet 数据集过滤；
- Query 数据集过滤的SQL语句；
- Op 运算符 ， Type为Field时有效，取值为AND,OR 默认为AND；
- Condition 过滤项的使用条件；
- FilterDependency 过滤条件依赖的组件的集合用","将字段隔开；

过滤值(FilterValue)属性

- FieldKey 值过滤字段，为字典的属性字段；
- RefValue 取值关联的字段；

- ParaValue 结果集过滤SQL中的参数；
- Type 过滤值的取值类型，属性如下：
  - Field 来源于字段，值根据RefValueKey指定的组件或单元格确定；
  - Formula 表达式，值来源于ParaValue定义的表达式；
  - Const 常量，值来源于ParaValue定义的常量。

  默认值为Formula。

17.4.14:动态字典(DynamicDict)

```
<DynamicDict RefKey="" AllowMultiSelection="" Independent="">
    <过滤>
</DynamicDict>
```

属性

- RefKey 动态字典ItemKey的关联字段， 取ItemKey时， 需保证RefKey字段对应的值正确；
- AllowMultiSelection - 是否允许多选；
- Independent 父子节点是否联动，true的时候不联动， false的时候联动 默认为false；

过滤条件同Dict，但在必要时需要为每个需要过滤的字典项设置过滤条件。根据ItemFilter中的ItemKey加以区分。

17.4.15:复合字典(CompDict)

```
<CompDict ItemKey="" DataItemKeys="" AllowMultiSelection="" Independent="">
    <过滤>
</CompDict>
```

属性

- ItemKey 字典对象的标识；
- DataItemKeys 复合字典的数据所在的字典；如果多个字典均可以作为选择数据，那么以逗号分隔；
- AllowMultiSelection 是否允许多选；
- Independent 父子节点是否联动，true的时候不联动， false的时候联动 默认为false；

过滤条件同Dict，但在必要时需要为每个需要过滤的字典项设置过滤条件。根据ItemFilter中的ItemKey加以区分。

17.4.16:日期选择(DatePicker)

```
<DatePicker Format="" OnlyDate="" NoTimeZone="">
</DatePicker>
```

- Format 时间的显示格式，默认值为yyyy-MM-dd，作为头控件使用时，该属性无效；
- OnlyDate 是否仅保留日期部分，默认值为True；
- NoTimeZone 是否无时区时间，目前未获支持

17.4.17:下拉框(ComboBox)

```
<ComboBox  Editable="" DynamicItems="" ItemsDependency="" SourceType="">
```

```
    <FormulaItems>
        <![CDATA[  ]]>
    </FormulaItems>
    <QueryDef>
        <Statement>
            <![CDATA[  ]]>
        </Statement>
        <ParameterCollection>
            <Parameter/>
            ...
        </ParameterCollection>
    </QueryDef>
     <Item Key="" Caption="" Value=""/>
     <Item Key="" Caption="" Value=""/>
     ... ...
     <Item Key="" Caption="" Value=""/>
     ... ...
</ComboBox>
```

下拉框属性

- Editable 是否可编辑；
- DynamicItems 是否是动态下拉项，取值为True或False，默认值为False；
- FormulaItems 脚本选项的下拉值；
- ItemsDependency 下拉项的依赖值的列表；
- SourceType 定义下拉项的来源，取值如下：
  - Items 来源于定义项目，即Item定义；
  - Formula 来源于表达式，FormulaItems定义表达式，其返回值以如下形式：
    - 字符串，以分号(;)分隔的字符串，每个部分以 [值,显示文字] 形式定义的，比如 "1,真;0,假"；
    - com.bokesoft.yes.common.struct.PairItemList，具体见API定义；
    - DataTable，其第0列包含值；
  - Query 来源于查询；返回的DataTable第0列包含值；

查询(QueryDef)属性值

- Statement 定义查询的内容；
- ParameterCollection 见查询参数集定义；

下拉项(Item)属性

- Key 项目的标识；
- Caption 项目的名称；
- Value 项目的值。

17.4.18:多选下拉框(CheckListBox)

```
<CheckListBox DynamicItems="" ItemsDependency="" SourceType="">
    <FormulaItems>
        <![CDATA[  ]]>
    </FormulaItems>
    <QueryDef>
        <Statement>
            <![CDATA[  ]]>
        </Statement>
        <QueryParameterCollection>
```

```
            <QueryParameter/>
            ...
        </QueryParameterCollection>
    </QueryDef>
     <Item Key="" Caption="" Value=""/>
     <Item Key="" Caption="" Value=""/>
     ... ...
     <Item Key="" Caption="" Value=""/>
     ... ...
</CheckListBox>
```

多选下拉框属性

- DynamicItems是否是动态下拉项，取值为True或False，默认值为False；
- FormulaItems 脚本选项的下拉值；
- ItemsDependency为下拉项的依赖值的列表；
- SourceType 定义下拉项的来源，取值如下：
  - Items 来源于定义项目，即Item定义；
  - Formula 来源于表达式，FormulaItems定义表达式，其返回值以如下形式：
    - 字符串，以分号(;)分隔的字符串，每个部分以 [值,显示文字] 形式定义的，比如 "1,真;0,假"；
    - com.bokesoft.yes.common.struct.PairItemList，具体见API定义；
    - DataTable，其第0列包含值；
  - Query 来源于查询；返回的DataTable第0列包含值；

查询(QueryDef)属性值

- Statement 定义查询的内容；
- ParameterCollection 见查询参数集定义；

下拉项(Item)属性

- Key 项目的标识；
- Caption 项目的名称；
- Value 项目的值。

17.4.19:图片(Image)

```
<Image SourceType="" Source="" Stretch="" Stretch="">
    <OnClick>
        <![CDATA[ ]]
    </OnClick>
</Image>
```

- SourceType 图片来源类型，取值为Data或Resource，默认值为Data；
- Source 在SourceType为Resource时，定义图片来源；
- OnClick 鼠标点击事件；

17.4.20:超链接(HyperLink)

```
<HyperLink>
```

```
    <OnClick>
        <![CDATA[  ]]
    </OnClick>
</HyperLink>
```

- OnClick 点击事件；

17.4.21:表格(Grid)

```
<Grid NewEmptyRow="" IsDynamicColumnExpand="" HideGroup4Editing="" PageLoadType="" PageRowCount="">
    <RowClick>
        <![CDATA[  ]]
    </RowClick>
    <RowDblClick>
        <![CDATA[  ]]
    </RowDblClick>
    <FocusRowChanged>
        <![CDATA[  ]]
    </FocusRowChanged>
    <GridColumnCollection>
        <GridColumn Key="" Caption="" Width="" Sortable="" Visible="" Enable="" ColumnType="" ExpandKey="" ColumnExpand="" >
            <ColumnExpand ExpandType="" ExpandSourceType="" ColumnKey="">
                <![CDATA[  ]]
            </ColumnExpand>
            [NestColumns:
            <GridColumnCollection>
            </GridColumnCollection>
            ]
        </GridColumn>
        ...
    </GridColumnCollection>
    <GridRowCollection>
        <GridRow Key="" RowType="" RowHeight="" TableKey="" GroupKey="" GroupLevel="" DefaultLayer="">
            <UICheckRuleCollection>
                ...
            </UICheckRuleCollection>
            <GridCell Key="" Caption="" Enable="" CellType="" IsMerged="" IsMergedHead="" MergedRowSpan=""
 MergedColumnSpan=""
                CellGroupType="" IsSelect="" RowExpand="">
                <DataBinding />
                <RowExpand ExpandType="" Tag="" ItemKey="">
                    <![CDATA[  ]]
                </RowExpand>
                <RowGroup Tag="" ItemKey="" ItemMask="">
                    <![CDATA[  ]]
                </RowGroup>
            </GridCell>
            ...
        </GridRow>
        ...
    </GridRowCollection>
</Grid>
```

17.4.21.1:表格属性

```
<Grid NewEmptyRow="" IsDynamicColumnExpand="" HideGroup4Editing="" PageLoadType="" PageRowCount=""/>
```

- NewEmptyRow 在编辑状态下是否为明细生成一个空行；默认值为True；
- IsDynamicColumnExpand 是否动态列扩展，用作性能优化选项；
- HideGroup4Editing 编辑时是否隐藏分组行，默认值为True；
- FocusRowChanged 焦点行改变事件；
- RowClick 行点击事件；
- RowDblClick 行双击事件；
- PageLoadType 分页的类型，取值为DB(后台分页，即只加载一页数据)、UI(前台分页，加载全部数据，只显示一页)；
- PageRowCount 分页情况下每页的最大行数；

17.4.21.2:列集合(GridColumnCollection)属性

列集合由多个列(GridColumn)组成。

```
<GridColumnCollection>
    <GridColumn Key="" Caption="" Width="" Sortable="" Visible="" Enable="" ColumnType="" ExpandKey="" ColumnExpand="" >
        <ColumnExpand ExpandType="" ExpandSourceType="" ColumnKey="">
            <![CDATA[ ]]
        </ColumnExpand>
        [NestColumns:
        <GridColumnCollection>
        </GridColumnCollection>
        ]
    </GridColumn>
    ...
</GridColumnCollection>
```

17.4.21.2.1:列(GridColumn)属性

- Key 列的标识;
- Caption 列的名称;
- Width 列宽。取值为固定值(如30px)或比例(如50%)，默认值为固定值80px。
- Sortable 可排序，取值为True和False，默认值为True;
- Visible 可见性，取值为True和False，默认值为True;
- Enable 可用性，取值为True和False，默认值为True。
- ColumnType 列类型，取值如下:
  - Fix 固定列;
  - Detail 明细列;
  - Group 分组列;
  - Total 汇总列。
- ColumnExpand 是否列扩展，取值为True和False，默认为False;
- NestColumns 指的不是一个属性，表示在列中可以嵌套列集合，每个列只能嵌套一个。

17.4.21.2.2:列扩展(ColumnExpand)

- ExpandType 列扩展的类型，取值为Data(数据扩展)、Title(标题扩展)；默认值为Data;
- ExpandSourceType 列拓展的数据源类型,取值为data(来源于明细数据)和custom(来源于定制)；
- ColumnKey 列扩展标识;

17.4.21.3:行集合(GridRowCollection)

```
<GridRowCollection>
    <GridRow Key="" RowType="" RowHeight="" TableKey="TableKey" GroupKey="" GroupLevel="" DefaultLayer="">
        <UICheckRuleCollection>
        ...
        </UICheckRuleCollection>
        <GridCell Key="" Caption="" Enable="" CellType="" IsMerged="" IsMergedHead="" MergedRowSpan="" MergedColumnSpan=""
            CellGroupType="" IsSelect="" RowExpand="">
            <DataBinding />
            <RowExpand ExpandType="" Tag="" ItemKey="">
                <![CDATA[ ]]
            </RowExpand>
            <RowGroup Tag="" ItemKey="" ItemMask="">
                <![CDATA[ ]]
```

```
            </RowGroup>
        </GridCell>
        ...
    </GridRow>
    ...
</GridRowCollection>
```

行集合由个GridRow组成，每个GridRow由多个GridCell组成。

17.4.21.3.1:行(GridRow)属性

- Key 行标识；
- RowType 行类型，取值如下：
  - Fix 固定行；
  - Group 分组行；
  - Detail 明细行；
  - Total 汇总行。
- RowHeight 行高，默认值为30；
- TableKey 行绑定的数据表标识；
- GroupKey 所属分组的标识，只对分组行有效；
- GroupLevel 在树形分组下指定分组层次，保留属性；
- DefaultLayer 默认的数据层级，默认值为-1，即不区分层级；
- UICheckRuleCollection 行检查规则集合，见表单部分的说明。

17.4.21.3.2:单元格(GridCell)

```
<GridCell Key="" Caption="" Enable="" CellType="" IsMerged="" IsMergedHead="" MergedRowSpan="" MergedColumnSpan=""
    CellGroupType="" IsSelect="" RowExpand="">
    <DataBinding />
    <RowExpand ExpandType="" Tag="" ItemKey="">
        <![CDATA[ ]]
    </RowExpand>
    <RowGroup Tag="" ItemKey="" ItemMask="">
        <![CDATA[ ]]
    </RowGroup>
</GridCell>
```

单元格属性：

- Key 单元格的标识，整个表单内不重复；
- Caption 单元格的名称；
- Enable 单元格的可用性，表达式；
- CellType 单元格的类型，取值如下：
  - Label 标签；
  - Button 按钮；
  - TextEdit 文本编辑框；
  - TextButton 文本编辑框；
  - NumberEdit 数值编辑框；

  - CheckBox 复选框；
  - Dict 字典；
  - DatePicker 日期编辑框；
  - ComboBox 下拉框；
  - CheckListBox 多选下拉框；
  - Image 图片；
  - ImageList 图片列表；
  - HyperLink 超链接；

  默认值为Label。
- IsMerged 是否是合并区域中单元格；
- IsMergedHead 是否是合并区域中的最左顶单元格；
- MergedRowSpan 合并行的数量；
- MergedColumnSpan 合并列的数量；
- CellGroupType 单元格的分组类型：
  - None 无定义；
  - RowGroup 行分组；
  - RowTreeGroup 行树分组；

  默认值为None。
- IsSelect 是否选择列，只对类型为CheckBox的单元格有效；
- RowExpand - 是否行扩展，取值为True和False，默认为False；
- 其它组件属性，根据CellType而定。

行扩展属性RowExpand

- ExpandType 扩展类型，Plat(平面)、Tree(树形)；默认值为Plat；
- Tag 处理标识，取值为Dict(字典)及其它(先不定)；
- ItemKey Tag取值为Dict时用于定义字典标识；

行分组属性RowGroup

- Tag 处理标识，取值为Dict(字典)及其它(先不定)；默认值为空；
- ItemKey Tag取值为Dict时用于指定分组依据的字典标识；
- ItemMask 字典项目的掩码标识，后面再确定意义，目前无用处；

17.4.22:列表视图(ListView)

```
<ListView TableKey="" PageLoadType="" PageRowCount="">
    <RowClick>
```

```
        <![CDATA[  ]]>
    </RowClick>
    <RowDblClick>
        <![CDATA[  ]]>
    </RowDblClick>
    <FocusRowChanged>
        <![CDATA[  ]]>
    </FocusRowChanged>
    <ListViewColumnCollection>
        <ListViewColumn Key="" Width="" Caption="" ColumnType="" IsSequence="" DataColumnKey="" IsExpand="" DefaultValue=""
 DefaultFormulaValue="">
            <ExpandSource>
                <![CDATA[  ]]>
            </ExpandSource>
        </ListViewColumn>
        ...
    </ListViewColumnCollection>
</ListView>
```

列表视图属性：

- TableKey 绑字的数据源表；
- PageLoadType 分页类型，取值为DB(后台只加载一页数据)、UI(后台加载全部数据，界面只显示一页)；
- PageRowCount 每页最大数据行数；
- FocusRowChanged 焦点行改变事件；
- RowClick 行单击事件，脚本类型；
- RowDblClick 行双击事件，脚本类型；

列集合(ListViewColumnCollection)属性：

- Key 列的标识，整个Window内不重复；
- Caption 列的名称；
- DataColumnKey 列绑定的数据对象的列标识；
- ColumnType 列类型，取值如下：
  - Label 标签；
  - CheckBox 复选框；
  - HyperLink 超链接；
  - Button 按钮；
  - DatePicker 日期选择框；
  - Dict 字典；
  - ComboBox 下拉框；
  - NumberEdit 数值编辑框；
  - TextEdit 文本编辑框；
- Width 列宽，取值为固定值(75px)或比例值(50%)，默认值为固定值75px；
- DefaultValue 默认值；
- DefaultFormulaValue 默认表达式值；

- 其它属性，根据ColumnType定义的组件类型而定。

17.4.23:富文本编辑器(RichEditor)

```
<RichEditor>
</RichEditor>
```

无其它属性。

17.4.24:图表(Chart)

```
<Chart ChartType="" SourceType="" Title="" SeriesInRow="" SeriesAxisTitle="" CategoryAxisTitle="">
    <ChartDataSource>
        <Series DataKey="" Title="" SplitDataKey=""/>
        ...
        <Category DataKey=""/>
    </ChartDataSource>
</Chart>
```

图表属性

- ChartType 图表类型，取值如下：
  - HBar 水平条状图；
  - StackedHBar 累积水平条状图；
  - VBar 垂直条状图；
  - StackedVBar 累积垂直条状图；
  - Area 面积图；
  - Line 线图；
  - Scatter 散点图；
  - Pie 饼图；
- SourceType 数据的来源，目前取值只有"DataObject"；
- Title 图表的标题；
- SeriesAxisTitle 系列轴标题；
- CategoryAxisTitle 项目轴标题；

数据源-系列(Series)定义

- DataKey 系列的数据来源字段；
- Title 系列的标题；
- SplitDataKey 系列的拆分字段定义；

数据源-项目(Category)定义

- DataKey 项目的数据来源字段。

17.4.25:工具栏(ToolBar)

```
<ToolBar IsDefault="">
</ToolBar>
```

- IsDefault 是否默认工具栏标题，取值为True和False，默认值为False。

17.4.26:容器(Container)

```
<Container Style="" IsDefault="" DefaultFormKey="" FormulaFormKey="">
</Container>
```

- Style 样式，取值为栈式(Stack)和选项卡(Tab)。默认为Tab;
- DefaultFormKey 默认表单对象的标识，默认为空;
- FormulaFormKey 表达式定义的表单标识;
- IsDefault 是否默认容器，取值为True或False，默认为True;

17.4.27:内嵌子表单(EmbedSub)

```
<EmbedSub BindingGridKey="" LinkType="" SourceFields="" TargetFields=""/>
```

- BindingGridKey - 子表单关联的主表所在的表格的标识;
- LinkType 数据关联类型，取值为Parent(父对象关联)、Foreign(外键关联);
- SourceFields 在外键关联的情况下，定义源表的字段列表，以","分隔;
- TargetFields 在外键关联的情况下，定义目标表的字段列表，以","分隔。

17.4.28:附件组件(Attachment)

```
<Attachment>
</Attachment>
```