Fc_G

E10脚本使用指南

适用于开发人员、顾问、客户mis在使用建档、凭证设计器、在线建档设计器等其他设计器中开发脚本逻辑时使用

**目录**

[概述 5](#_Toc12219)

[目标 5](#_Toc575)

[e10中的脚本注意事项 6](#_Toc29837)

[基本类型 6](#_Toc18439)

[字符串 6](#_Toc22170)

[使用单引号(')创建字符串 6](#_Toc14153)

[使用双引号(")创建字符串 7](#_Toc14777)

[字符中包含单引号，双引号 8](#_Toc10212)

[字符串函数 9](#_Toc5150)

[str(x) 其他类型转换为字符 9](#_Toc9645)

[hex(x) 整形转换为十六进制字符串 9](#_Toc16122)

[oct(x) 整数转换为一个八进制字符串 9](#_Toc22292)

[len(str)换取字符串长度 9](#_Toc14582)

[cmp(x, y)比较2个值 9](#_Toc21038)

[通过下标做字符串截取 10](#_Toc20891)

[在目标中查找指定的字符targetStr.find(findstr)   11](#_Toc6918)

[在目标中查找指定的字符targetStr.rfind(findstr)  12](#_Toc11494)

[计算指定字符出现的次数str.count(substr)    12](#_Toc7505)

[替换为指定的字符串str.replace(oldstr, newstr)    13](#_Toc18588)

[去掉前后的空格S.strip()  13](#_Toc17669)

[去掉左边的空格S.lstrip()  13](#_Toc14618)

[去右边的空格S.rstrip()  13](#_Toc13409)

[布尔类型 13](#_Toc16811)

[数字 14](#_Toc19295)

[数学函数 15](#_Toc19642)

[abs(x)绝对值 16](#_Toc5045)

[round(x [,n]) 四舍五入 16](#_Toc10832)

[cmp(x, y)比较2个值 16](#_Toc32199)

[max(x1, x2,...)求最大值 16](#_Toc28606)

[min(x1, x2,...) 求最小值 16](#_Toc32310)

[pow(x, y) 求几次方 16](#_Toc26984)

[int(x ) 将x转换为一个整数 16](#_Toc22776)

[float(x ) 将x转换到一个浮点数 17](#_Toc22532)

[sum(x1,x2,...) 求集合中指定属性的汇总值 18](#_Toc3611)

[Average(collection,x) 求集合中指定属性的平均值 19](#_Toc15572)

[Count(collection) 求集合中元素的数目 19](#_Toc15771)

[数组（元组） 19](#_Toc21663)

[创建数组 19](#_Toc17143)

[访问数组 20](#_Toc8050)

[数组函数 20](#_Toc3691)

[cmp(tuple1, tuple2) 比较两个数组元素。 20](#_Toc883)

[len(tuple) 计算数组元素个数。 21](#_Toc12534)

[max(tuple) 返回数组中元素最大值。 21](#_Toc19276)

[min(tuple) 返回元组中元素最小值。 21](#_Toc31127)

[in 判断是否存在数组中 21](#_Toc1279)

[其他常用函数 21](#_Toc13667)

[ConvertToInt32(object alue) 转换数字类的值为 Int32 21](#_Toc11280)

[Decimal(object value) 转换数字类的值为 deciaml 21](#_Toc29031)

[String(object value) 将对象转换为字符串 21](#_Toc8825)

[IsNull(object value) 22](#_Toc21614)

[IsNotNull(object value) 22](#_Toc21953)

[GetTimeDiff(DateTime dt1, DateTime dt2, int type) 22](#_Toc6179)

[AddTimeValue(DateTime dt,int type, int value) 22](#_Toc6688)

[条件判断 23](#_Toc9261)

[相等运算 23](#_Toc21419)

[条件分支 23](#_Toc9009)

[使用场景 24](#_Toc1486)

[语法 24](#_Toc26848)

[案例 24](#_Toc25500)

[嵌套条件分支语法 24](#_Toc31101)

[案例 25](#_Toc4044)

[访问界面上的实体字段 25](#_Toc18041)

[什么是ActiveObject 27](#_Toc29009)

[使用ActiveObject访问当前实体字段 27](#_Toc4382)

[使用ActiveObject.Parent访问父实体字段 28](#_Toc27115)

[使用ActiveObject.[单身名字]访问单身集合 29](#_Toc11100)

[什么是PreviousObject 31](#_Toc26159)

[访问界面上的实体关联的字段（sharedata） 31](#_Toc20534)

[访问服务 33](#_Toc15056)

[获取系统日期时间 34](#_Toc22627)

[根据用户自定义的查询去数据库查询数据，如果查询到数据，返回True，否则返回False 34](#_Toc8500)

[根据用户自定义的查询去数据库查询数据，如果查询到数据，返回True，否则返回False 35](#_Toc29977)

[获取数据库中的字段值 36](#_Toc30096)

[返回主表的字段值 36](#_Toc10789)

[返回子表的指定字段值 37](#_Toc11052)

[登录服务。提供登录用户的相关信息 38](#_Toc18866)

[根据Oid检查数据是否存在，适用于主实体 38](#_Toc20349)

[根据属性名检查数据是否存在，适用于主实体 39](#_Toc1912)

[根据Oid检查数据是否存在，根据Path不同查询主实体或明细实体 40](#_Toc31528)

[访问系统参数 40](#_Toc9763)

[用户参数 41](#_Toc18495)

[系统全局参数 43](#_Toc6452)

# 概述

脚本是e10中非常重要的部分，不掌握脚本，就没有办法写e10的业务，也不能扩展e10的功能。而脚本作为一种编程语言，有一定的难度，需要一些时间的学习才能掌握，本指南主要引导没有编程基础的人员使用脚本，并提供抛砖引玉的作用。

本指南需要用到e10，请保证本地有可以运行的e10的客户端即可。

# 目标

通过翻阅或者学习该指南，可以了解、掌握如何在e10的开发设计器、在线设计器等其他设计器中使用脚本。

* 简单的脚本语法介绍，包括四则运算、条件分支（if else）（10min）
* 字符串的相关函数、数组的相关函数（30min）
* 在脚本中访问界面上的字段、关系（sharedata）（20min）
* 在脚本中访问服务，会使用访问数据库字段的服务、判断数据存在的服务，以及目前的服务列表（30min）
* 在脚本中访问系统参数、用户参数，以及目前e10中系统参数列表（30min）

# e10中的脚本注意事项

E10中只使用到Python脚本中很少的一块，就是短句：

1. 短句是有且仅有的一条可执行的语法，类似于一条可执行的sql，不能是多条语句，也不能和sql一样定义参数；
2. 可以返回一个值或者不返回任何值；

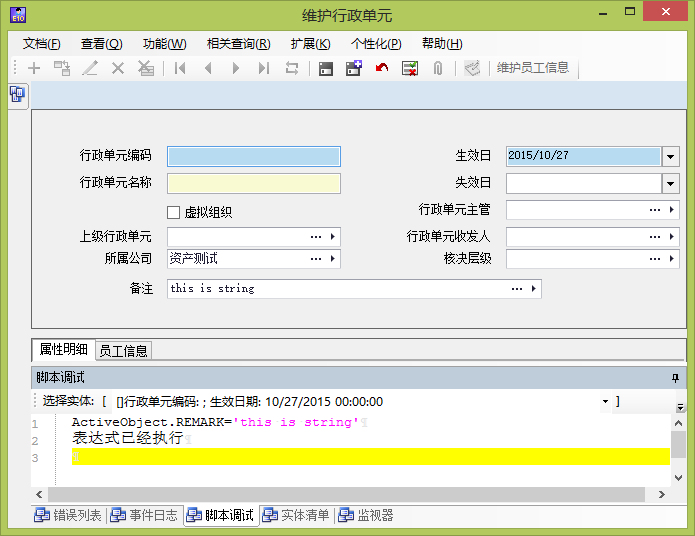
# 基本类型

## 字符串

如何在Python中使用字符串

### 使用单引号(')创建字符串

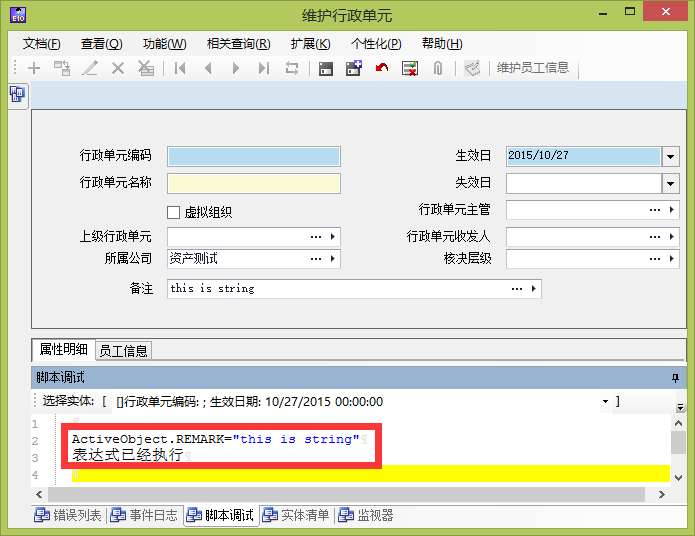
用单引号括起来表示字符串，例如：  
'this is string'

测试范例（**注，所有测试范例都是在客户端使用调试模式启动，使用“脚本调试“面板进行的**）：  


### 使用双引号(")创建字符串

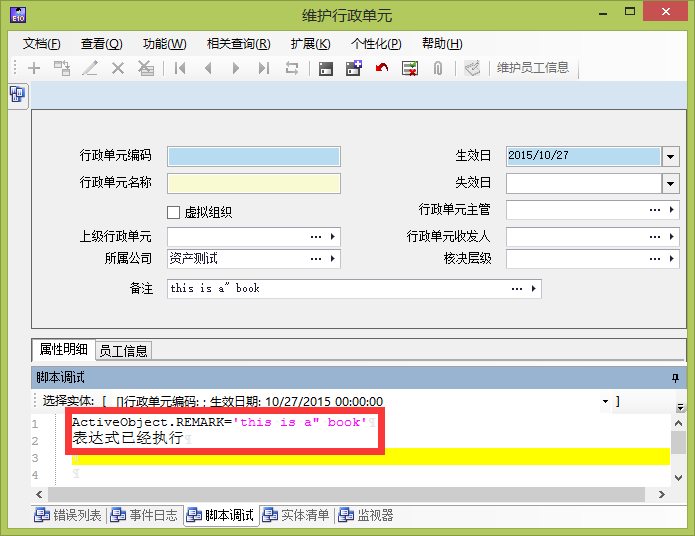
双引号中的字符串与单引号中的字符串用法完全相同，例如：  
"this is string"

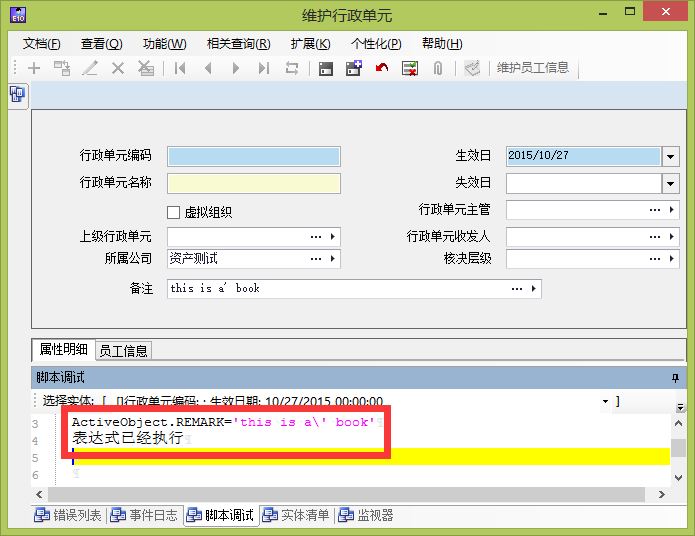
测试范例：



### 字符中包含单引号，双引号

'this is a" book'  
'this is a"" book'   
'this is a\' book'



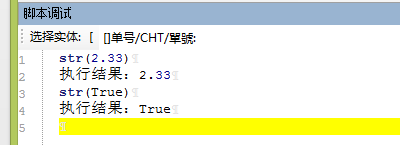


## 字符串函数

### str(x) 其他类型转换为字符

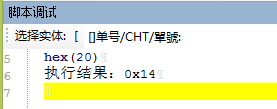
将对象x转换为字符串

测试范例：



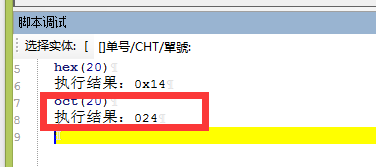
### hex(x) 整形转换为十六进制字符串

将一个整数转换为一个十六进制字符串



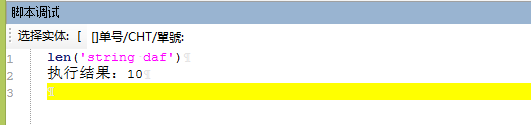
### oct(x) 整数转换为一个八进制字符串

将一个整数转换为一个八进制字符串



### len(str)换取字符串长度

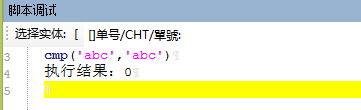
测试范例：



### cmp(x, y)比较2个值

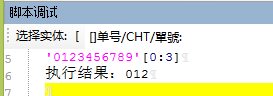
如果 x < y 返回 -1, 如果 x == y 返回 0, 如果 x > y 返回1

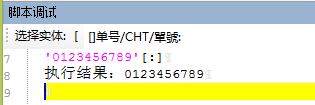
测试范例：



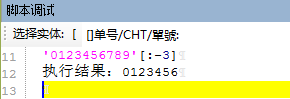
### 通过下标做字符串截取

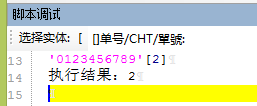
**特别注意：下标从0开始；str[0:3]不包含下标为3的字符。**  
str = ‘0123456789’  
str[0:3] #截取第一位到第三位的字符

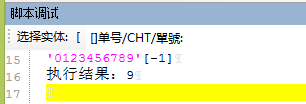
  
str[:] #截取字符串的全部字符

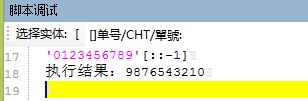
  
str[6:] #截取第七个字符到结尾

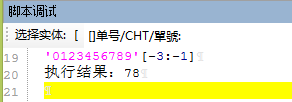
  
str[:-3] #截取从头开始到倒数第三个字符之前

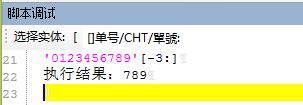
  
str[2] #截取第三个字符

  
str[-1] #截取倒数第一个字符

  
str[::-1] #创造一个与原字符串顺序相反的字符串

  
str[-3:-1] #截取倒数第三位与倒数第一位之前的字符

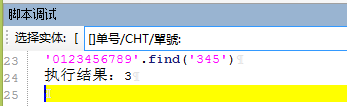
  
str[-3:] #截取倒数第三位到结尾



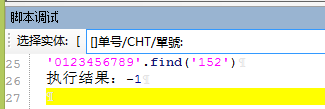
### 在目标中查找指定的字符targetStr.find(findstr)

#返回targetStr中出现findstr的第一个字母的标号，如果targetStr中没有findstr则返回-1。

测试范例：



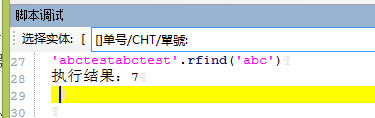
测试范例：



### 在目标中查找指定的字符targetStr.rfind(findstr)

#返回S中最后出现的substr的第一个字母的标号，如果S中没有substr则返回-1，也就是说从右边算起的第一次出现的substr的首字母标号

测试范例：

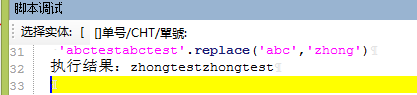


### 计算指定字符出现的次数str.count(substr)

#计算substr在S中出现的次数

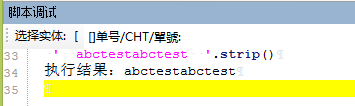
### 替换为指定的字符串str.replace(oldstr, newstr)

#把str中的oldstr替换为newstr是替换的通用形式，还有一些函数进行特殊字符的替换   
测试范例：

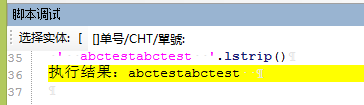


### 去掉前后的空格S.strip()

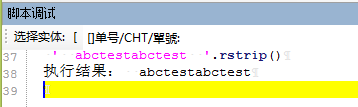
测试范例：



### 去掉左边的空格S.lstrip()



### 去右边的空格S.rstrip()

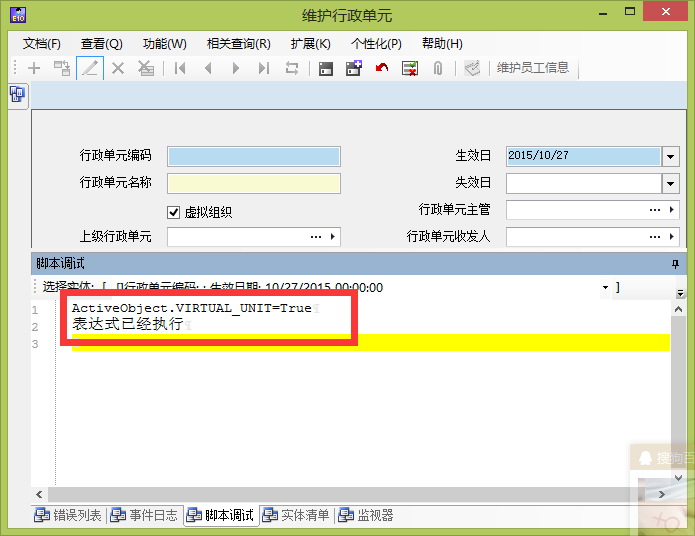


## 布尔类型

False（表示假）,True（表示真）

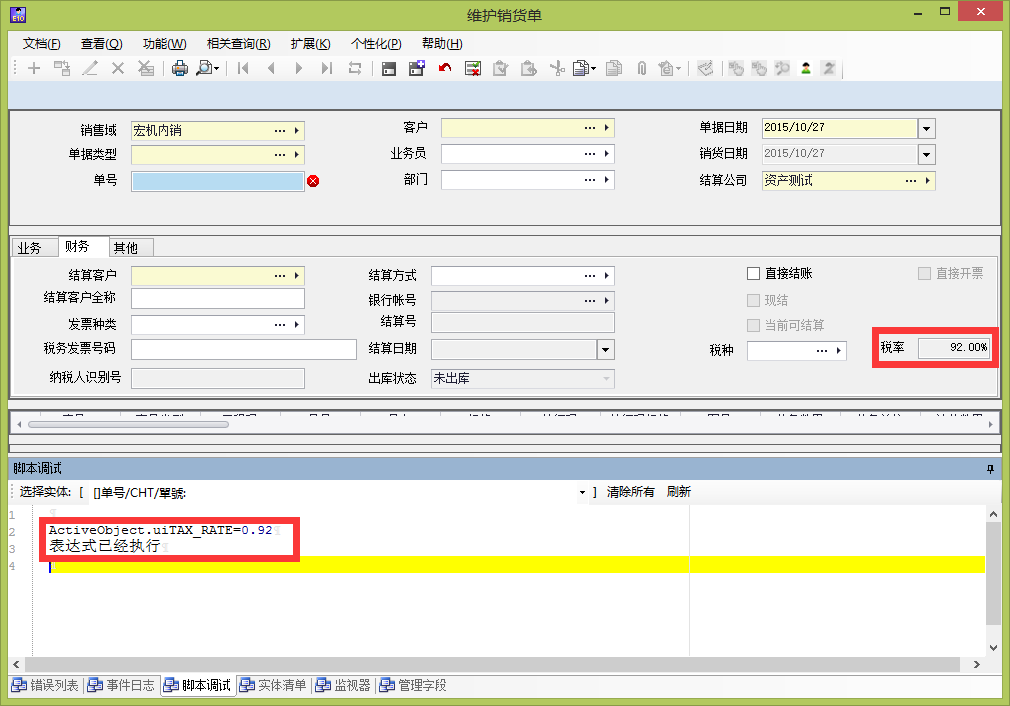
脚本中区分大小写，所以一定要注意首字母是大写的，写成true，false都不是正确的布尔类型

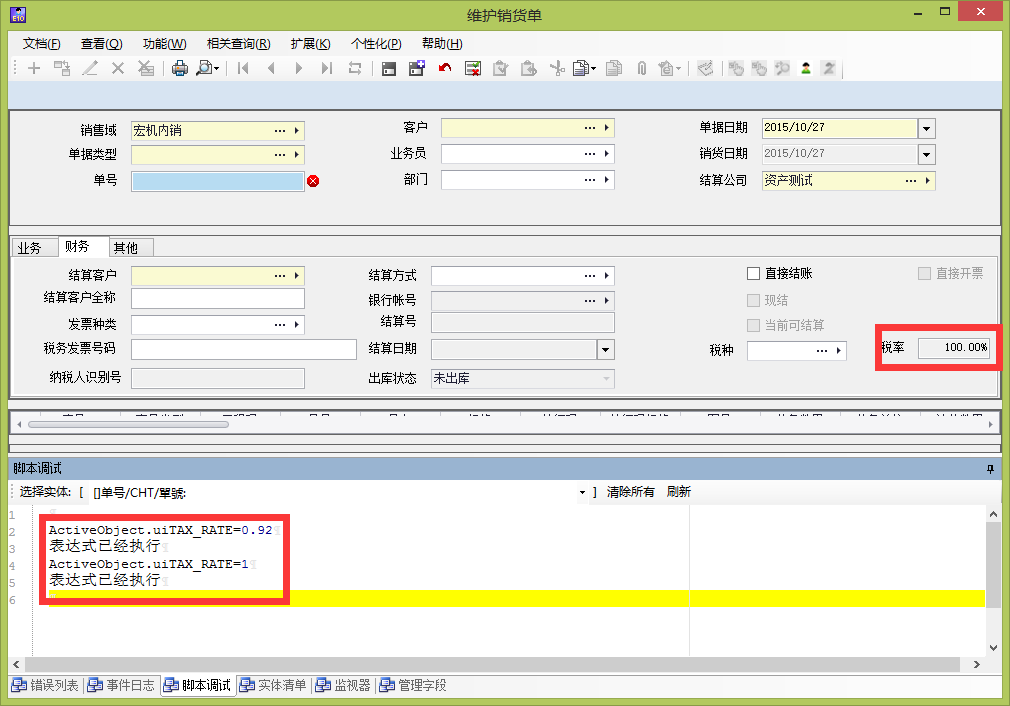
测试范例：



## 数字

脚本中有整形数字和浮点数。





## 数学函数

### abs(x)绝对值

返回数字的绝对值，如abs(-10) 返回 10

### round(x [,n]) 四舍五入

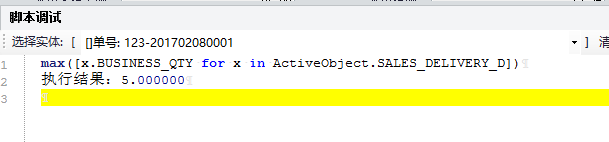
返回浮点数x的四舍五入值，如给出n值，则代表舍入到小数点后的位数。

### cmp(x, y)比较2个值

如果 x < y 返回 -1, 如果 x == y 返回 0, 如果 x > y 返回 1

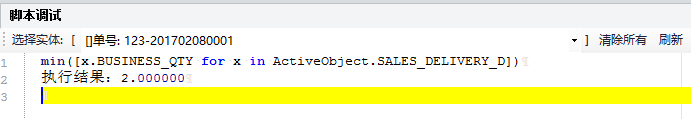
### max(x1, x2,...)求最大值

返回给定参数的最大值，参数可以为序列。



### min(x1, x2,...) 求最小值

返回给定参数的最小值，参数可以为序列。



### pow(x, y) 求几次方

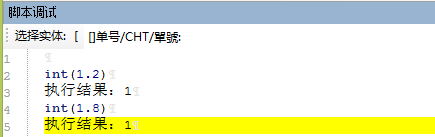
x\*\*y 运算后的值。

### int(x ) 将x转换为一个整数

把浮点数转换为整数，注意，这里会把小数点后面的值全部清除掉。

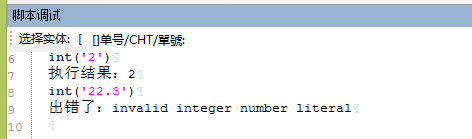
例如int(1.2)返回1，int(1.8)也返回1

测试范例：



我们也可以通过这个函数把字符串转换为int，不过字符串必须是表示整形，例如int('2')返回2，int('22.3')则报错，因为字符串’22.3’只能转换为浮点数，而不是整数

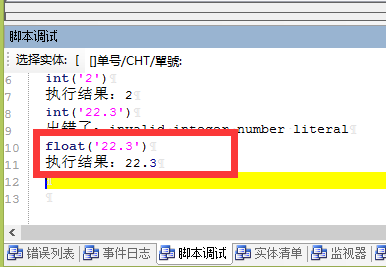
测试范例：



### float(x ) 将x转换到一个浮点数

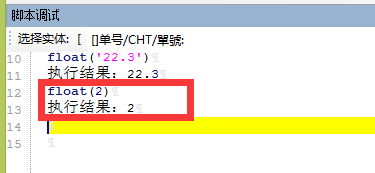
可以把其他类型转换为浮点数，最常用的是把字符串转换为浮点数

测试范例：



把int转换为浮点数

测试范例：



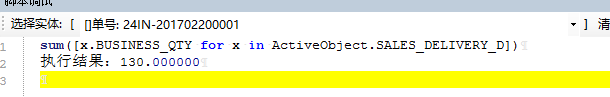
### sum(x1,x2,...) 求集合中指定属性的汇总值

求集合中指定属性的汇总值，返回集合中指定属性的汇总值

测试范例：

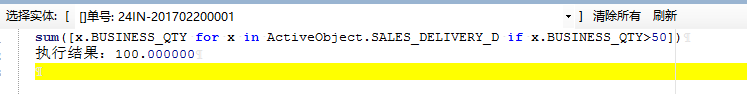
如：求销货单的单身业务数量的总计

推荐方式：



根据指定条件求汇总值：

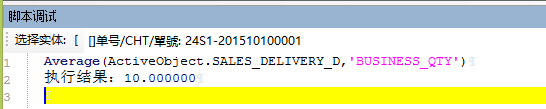
求销货单单身业务数量大于50的总计：



### Average(collection,x) 求集合中指定属性的平均值

求集合中指定属性的平均值，返回集合中指定属性的平均值，x：明细的属性，可以使用复杂属性

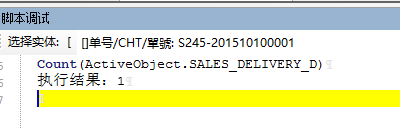
求销货单的单身业务数量的平均值：



### Count(collection) 求集合中元素的数目

求集合中元素的数目，返回集合中元素的数目

求销货单单身的总笔数：



## 数组（元组）

### 创建数组

数组创建很简单，只需要在括号中添加元素，并使用逗号(,)隔开即可，例如：

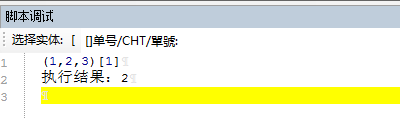
('physics', 'chemistry', 1997, 2000)

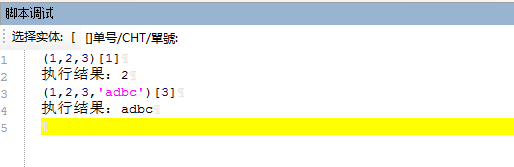
(1, 2, 3, 4, 5 )

数组中只有一个元素时，需要在元素后面添加逗号，例如： (50,);

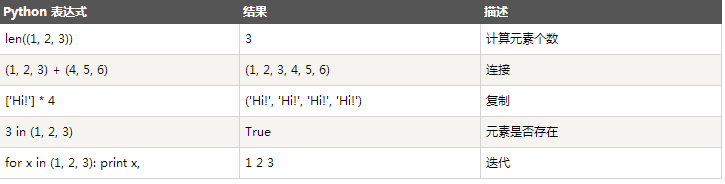
### 访问数组

数组与字符串类似，下标索引从0开始，可以进行截取，组合等。





### 数组函数

与字符串一样，数组之间可以使用+号和\*号进行运算。这就意味着他们可以组合和复制，运算后会生成一个新的数组。  


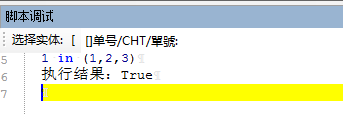
### cmp(tuple1, tuple2) 比较两个数组元素。

### len(tuple) 计算数组元素个数。

### max(tuple) 返回数组中元素最大值。

### min(tuple) 返回元组中元素最小值。

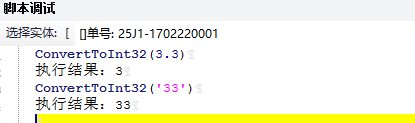
### in 判断是否存在数组中



## 其他常用函数

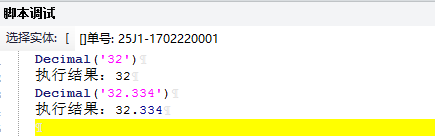
### ConvertToInt32(object alue) 转换数字类的值为 Int32

将value值(数字类的值)转换为Int32类型的值对象，返回Int32值对象



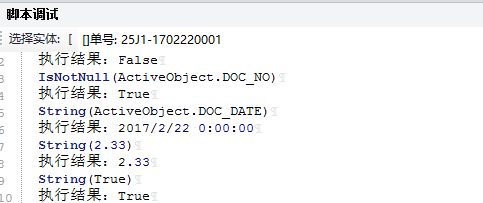
### Decimal(object value) 转换数字类的值为 deciaml

将value值(数字类的值)转换为decimal类型的值对象，返回 decimal 值对象



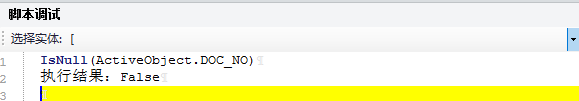
### String(object value) 将对象转换为字符串

将对象转换为字符串(如果对象为 null，则返回空字符串而不是null)，返回对象的字符串表示形式



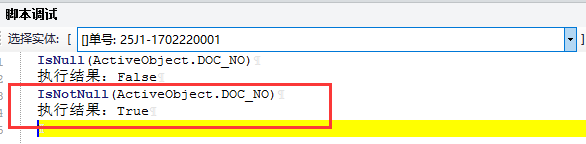
### IsNull(object value)

检查值是否为 null 或 DBNull.Value,如果对象为 null 或 DBNull.Value，则返回真



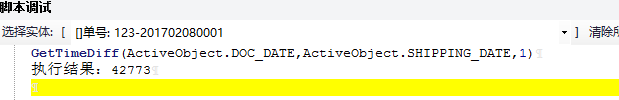
### IsNotNull(object value)

检查对象是否不为 null 并且 不为 DBNull.Value,如果对象不为 null 并且不为 DBNull.Value，则返回真



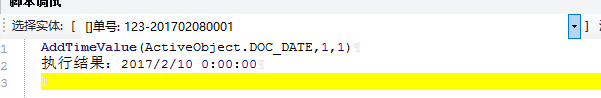
### GetTimeDiff(DateTime dt1, DateTime dt2, int type)

返回第一个日期时间减去第二个日期时间的值，dt1:第一个日期时间，dt2:第二个日期时间，type:返回类型， 1：间隔天数；2：间隔小时数；3：间隔分钟数；4：间隔秒数



### AddTimeValue(DateTime dt,int type, int value)

在日期时间上增加指定类型的值,dt:要增加类型的日期时间值，type:返回类型， 1：间隔天数；2：间隔小时数；3：间隔分钟数；4：间隔秒数,value:增加的值，为负数时表示是减值,返回增加值后的日期时间



## 条件判断

### 相等运算

==，and，or，True,False

==，用来判断2个值是否相等，相等返回True，否则返回False

例如： ActiveObject.TVT\_RESTRICTIVE=='0' （ps：目的类型性质约束为0）

and, 逻辑与，两个条件必须都为True，返回True，否则返回False

例如：ActiveObject.uiUNIT\_MODE=='S' and ActiveObject.uiPACKING\_UNIT==True

or, 逻辑与，两个条件只要有一个为True，返回True，都为False返回False

例如：当单据类型核对订单**或**内部交易勾选时

ActiveObject.uiCHECK\_SALES\_ORDER==True or ActiveObject.uiINTERNAL==True

True，脚本中固定值，表示真，大小写必须一致；

False，脚本中固定值，表示假，大小写必须一致

### 条件分支

### 使用场景

**如果** 条件成立 则 执行逻辑或者返回值A

否则 执行逻辑或者返回值B

### 语法

执行逻辑或者返回值A if 条件 else执行逻辑或者返回值B

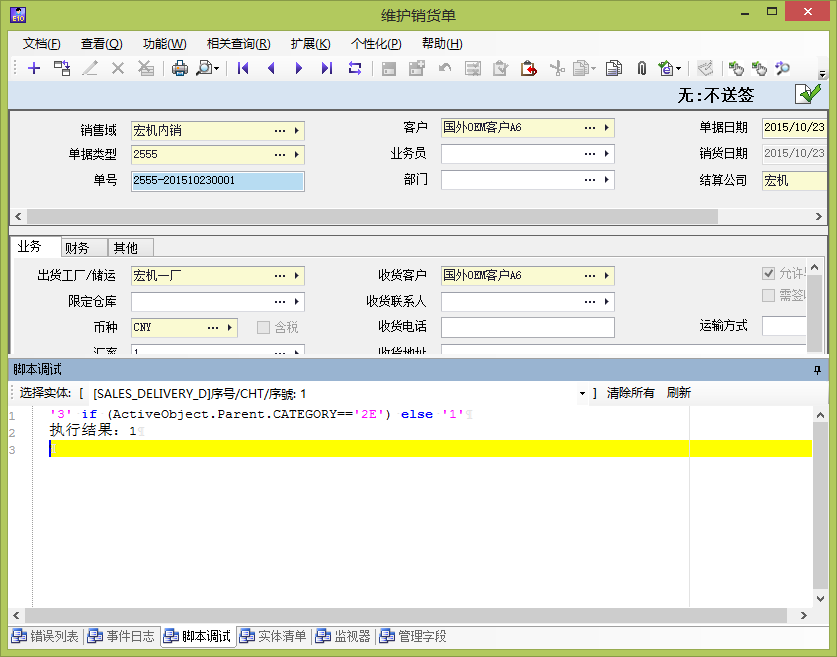
### 案例

销货单中要处理逻辑：

如果单头【单据性质】等于 寄售销货单（2E），则单身【销货类型】=【寄售销货】(3),否则单身【销货类型】=【一般销货】(1)

表达式的写法：

'*3' if (ActiveObject.Parent.CATEGORY=='2E') else '1'*



### 嵌套条件分支语法

执行逻辑或者返回值A if 条件2 else执行逻辑或者返回值B if 条件1 else执行逻辑或者返回值C if 条件3 else执行逻辑或者返回值D

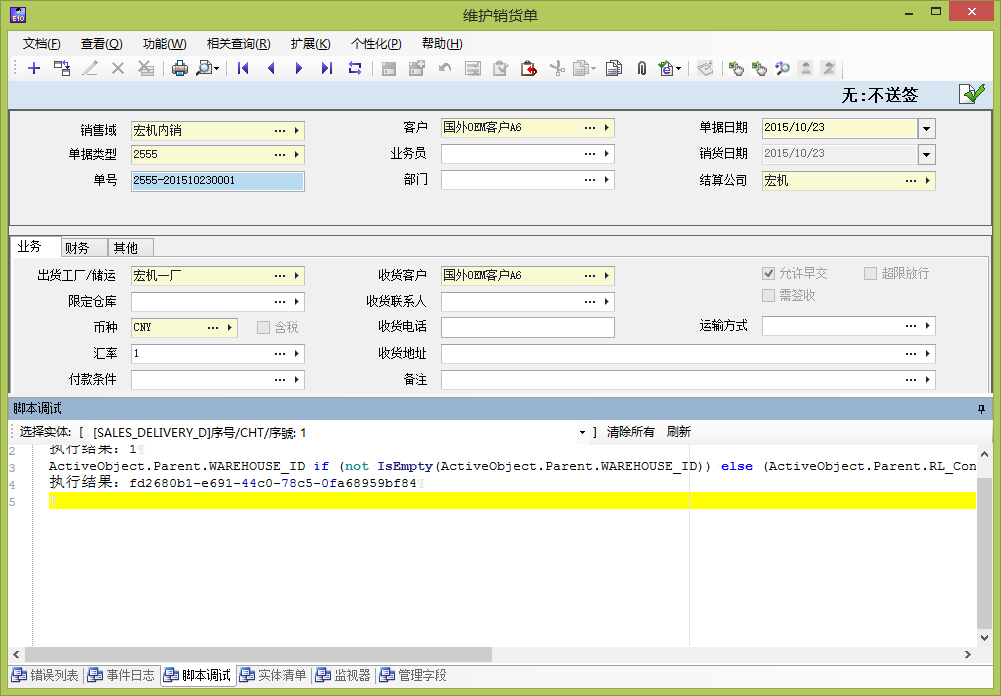
### 案例

销货单中要处理逻辑：

当单头限定仓库不为空时，默认单头限定仓库,否则默认客户寄售关系的寄售仓

表达式的写法：

*ActiveObject.Parent.WAREHOUSE\_ID if (not IsEmpty(ActiveObject.Parent.WAREHOUSE\_ID)) else (ActiveObject.Parent.RL\_Consign.WAREHOUSE\_ID if ActiveObject.Parent.CATEGORY=='2E' else ActiveObject.WAREHOUSE\_ID)*



## 访问界面上的实体字段

在脚本中我们需要访问当前界面上的实体字段来做业务处理，必须使用关键字ActiveObject

### 什么是ActiveObject

ActiveObject是E10的一个特殊关键字，代表当前界面正在编辑的数据。它可能代表单头、单身的一行数据或者子单身的一行数据。

那ActiveObject什么时候代表单头，什么时候代表单身的一行数据，或什么时候代表子单身的一行数据？

当我们的执行逻辑是针对单头时，比如单头字段赋值，单头校验等，那么ActiveObject代表单头。

当我们的鼠标选中单身一行时，要对这一行的执行业务逻辑，那么ActiveObject代表单身的这一行；

对子单身同理。

案例：

1. 在销货单我们要汇总单身件数，并显示到单头字段【PIECES】，我们需要写如下汇总公式：

Sum(ActiveObject.SALES\_DELIVERY\_D,"PIECES")

**因为我们的目标字段是单头**，所以ActiveObject代表单头，这里SALES\_DELIVERY\_D代表了所有的单身

1. 在销货单单身每新增一笔时，单身的销货类型根据单头来获取值：

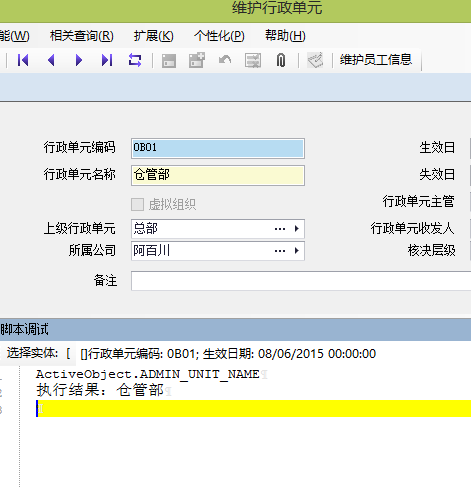
'3' if (ActiveObject.Parent.CATEGORY=='2E') else '1'

因为**我们要给单身字段赋值**，那么ActiveObject代表了单身当前聚焦行，也就是我们的新增行。这里Parent代表单头

### 使用ActiveObject访问当前实体字段

通过【ActiveObject.字段名】的方式我们可以访问当前实体上的字段；

案例：

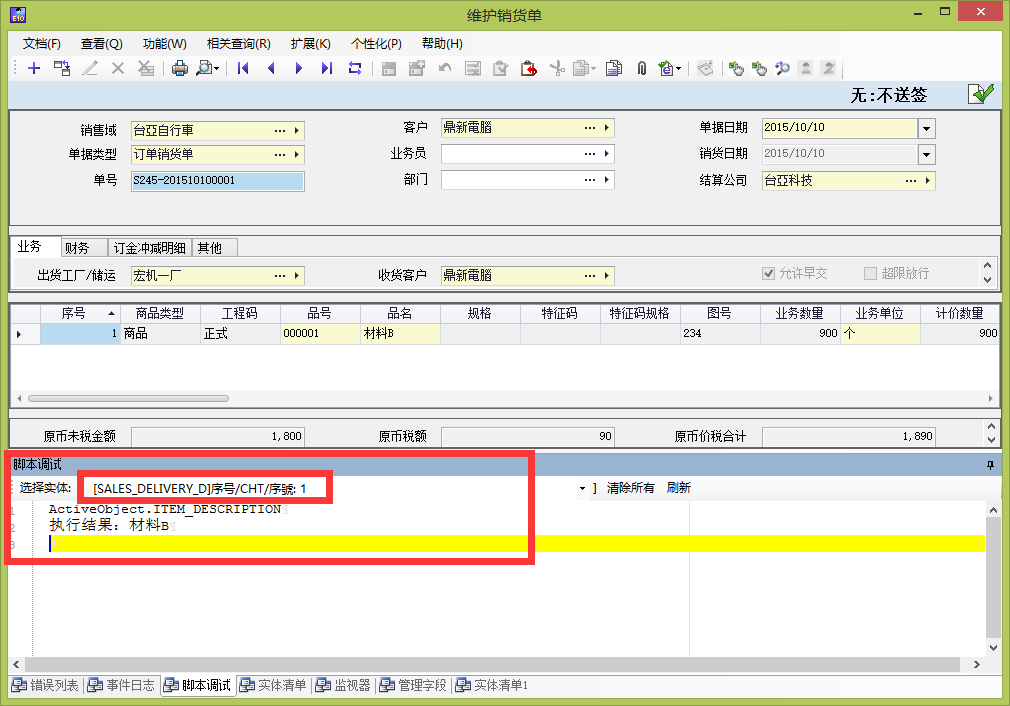


### 使用ActiveObject.Parent访问父实体字段

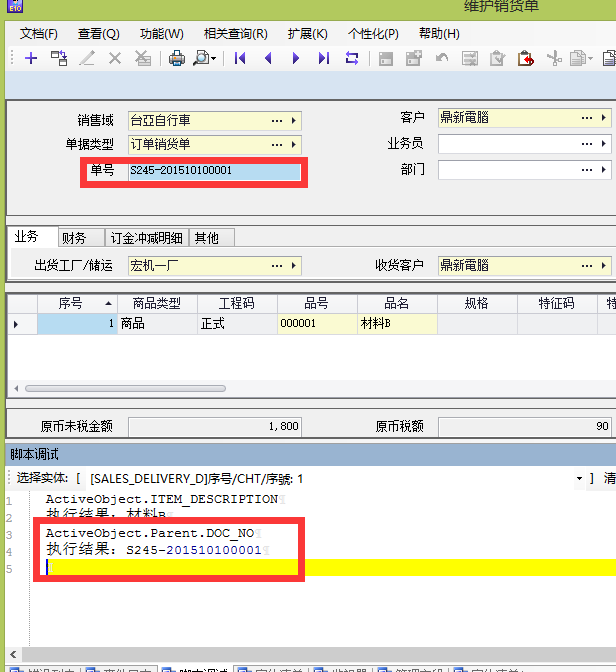
如果当前ActiveObject是在单身，我们可以通过【ActiveObject.Parent.字段名】来访问**单头的字段**；

如果当前ActiveObject是在子单身，我们可以通过【ActiveObject.Parent.字段名】来访问**单身的字段**，通过【ActiveObject.Parent.Parent.字段名】来访问**单头的字段**。

案例：访问单身字段



例：在单身访问单头字段



### 使用ActiveObject.[单身名字]访问单身集合

如果当前ActiveObject有单身为DETAIL\_D，我们可以通过【ActiveObject. DETAIL\_D】来访问集合，返回的是单身全部数据。

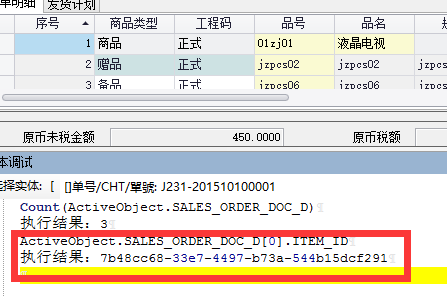
比如在订单单头，可以通过ActiveObject.SALES\_ORDER\_DOC\_D获取全部数据

案例：

可以通过Count(ActiveObject.SALES\_ORDER\_DOC\_D)得到单身的笔数

案例：

可以通过ActiveObject.SALES\_ORDER\_DOC\_D[0]得到第一个笔数据；



可以通过ActiveObject.SALES\_ORDER\_DOC\_D[0]. SALES\_ORDER\_DOC\_SD得到第一笔单身的所有子单身

### 什么是PreviousObject

我们在连续新增单据的时候，或者连续新增单身行的时候，可以通过PreviousObject来获取上一次新增的单据或者上一笔单身记录。

PreviousObject的使用方式和ActiveObject一样，**但是在公式中只能使用连续新增时机点!**



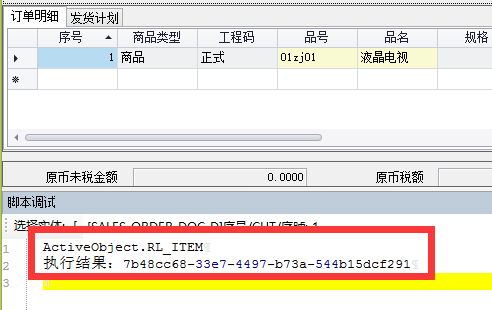
## 访问界面上的实体关联的字段（sharedata）

大部分情况下，我们除了需要获取当前界面上的字段之外，我们还要获取关联表的字段。

访问格式：

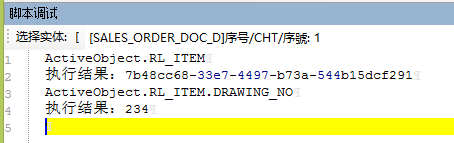
* ActiveObject.RL\_[关联实体名]

关联的实体名必须是当前实体表上之间关联的表，比如【订单】单头上关联了表【单据类型】



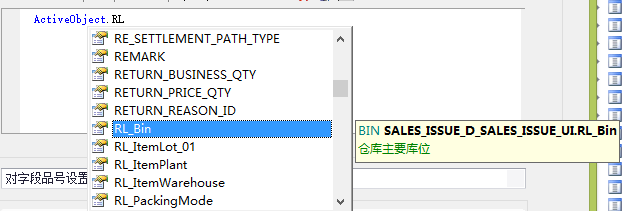
* ActiveObject.RL\_[关联实体名].字段名

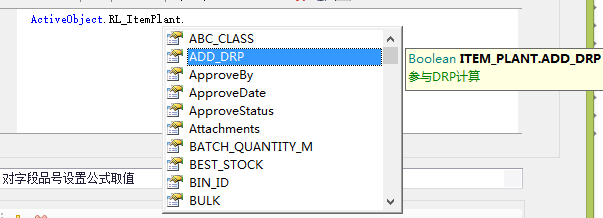
通过关联访问关联表的字段，如下：



* ActiveObject.[自定义的关联名]

我们也可以通过自定义的关联来访问想要的字段，这里有2种方式去知道定义了哪些关联：

1. 通过查询配置档的方式
2. 通过在线建档设计器（以及以后的新设计器）的脚本编写工具，自动列出所有的关联。



## 访问服务

E10提供了大量的服务来帮助我们在脚本中完成业务逻辑。

访问服务的方式如下：

**Context[目标typekey].服务名(参数列表)**

**或者**

**Context.服务名(参数列表)**

常用服务列表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **服务名称** |  | 获取系统日期时间 | | | | |
| **说明** |  | 自定义校验，判断是否数据符合自定义的业务逻辑 | | | | |
| **函数** |  | Context.Now,Context. NowDate | | | | |
| **函数说明** |  | 在客户端如果要获取系统时间，请使用此函数 | | | | |
| **参数说明** |  |  | | |  | |
| **返回值** |  | 返回服务器系统日期和时间 | | | | |
|  |  |  | | | | |
| **服务名称** |  | **sql校验** | | | | |
| **说明** |  | **自定义校验，判断是否数据符合自定义的业务逻辑** | | | | |
| **函数** |  | **Context.SqlValidate(‘sql’,参数列表，参数值列表)** | | | | |
| **函数说明** |  | 根据用户自定义的查询去数据库查询数据，如果查询到数据，返回True，否则返回False 注意：在大版中请勿使用，大版中请使用OoqlValidate，否则以后迁移到oracle数据库的时候出现不能预料的问题 | | | | |
| **参数说明** |  | sql : sql脚本  参数列表：定义的参数列表  参数值列表：定义了几个参数，就必须传入相应的参数值 | | |  | |
| **返回值** |  | 如果查询到数据，返回True，否则返回False | | | | |
| **范例** |  | 判断数据库中是否存在用户输入的工厂id且生效  Context.**SqlValidate**('select top 1 PLANT\_ID from PLANT Where PLANT\_ID=@ID and ApproveStatus=\'Y\'','ID',ActiveObject.PLANT\_ID) | | | | |
|  |  |  | | | | |
| **服务名称** |  | **Ooql校验** | | | | |
| **说明** |  | **自定义校验，判断是否数据符合自定义的业务逻辑** | | | | |
| **函数** |  | **Context.OoqlValidate(‘ooql’,参数列表，参数值列表)** | | | | |
| **函数说明** |  | 根据用户自定义的查询去数据库查询数据，如果查询到数据，返回True，否则返回False | | | | |
| **参数说明** |  | ooql : ooql脚本，ooql脚本的写法请参照ooql使用说明  参数列表：定义的参数列表  参数值列表：定义了几个参数，就必须传入相应的参数值 | | |  | |
| **返回值** |  | 如果查询到数据，返回True，否则返回False | | | | |
| **范例** |  | 判断数据库中是否存在用户输入的工厂id且生效  Context.**OoqlValidate**('OOQL.Select(1,"PLANT\_ID").From("PLANT").Where((OOQL.AuthFilter("PLANT","PLANT"))&((OOQL.CreateProperty("PLANT\_ID")== OOQL.CreateParameter("ID"))&(OOQL.CreateProperty("ApproveStatus")== OOQL.CreateConstants("Y"))))','ID',ActiveObject.PLANT\_ID) | | | | |
|  |  |  | | | | |
| **服务名称** |  | 获取数据库中的字段值 | | | | |
| **说明** |  | 获取数据库中的字段值 | | | | |
| **函数** |  | Object GetEntityProperty（返回字段，参数列表，参数值列表） | | | | |
| **函数说明** |  | 返回主表的字段值 | | | | |
| **参数说明** |  | 返回字段：需要查询回来的字段  参数列表：定义的参数列表  参数值列表：定义了几个参数，就必须传入相应的参数值 | | |  | |
| **返回值** |  | 返回数据库中指定字段的值 | | | | |
| **范例** |  | 根据doc\_id返回doc表中单据类型名称  Context["DOC"].GetEntityProperty("DOC\_NAME",('DOC\_ID','ApproveStatus'),(ActiveObject.DOC\_ID,'Y')) | | | | |
|  |  |  | | | | |
| **函数** |  | Object GetEntityProperty（path，返回字段，参数列表，参数值列表） | | | | |
| **函数说明** |  | 返回子表的指定字段值 此函数有效率问题，慎用！！！！！ | | | | |
| **参数说明** |  | Path，单身名  返回字段：需要查询回来的字段  参数列表：定义的参数列表  参数值列表：定义了几个参数，就必须传入相应的参数值 | | | | |  | |
| **返回值** |  | 返回数据库中指定字段的值 | | | | |
| **范例** |  | 返回销售中心表的单身的默认工厂值  Context["SALES\_CENTER"].GetEntityProperty("SALES\_CENTER\_PLANT","PLANT\_ID",('SALES\_CENTER\_ID','MAIN','ApproveStatus'),(V.DEF\_SALES\_CENTER\_ID['@'],True,'Y')) | | | | |
|  |  |  | | | | |
| **服务名称** |  | ILogOnService | | | | |
| **说明** |  | 登录服务。提供登录用户的相关信息 | | | | |
| **函数** |  | String CurrentUserId | | | | |
| **函数说明** |  | 返回当前登录用户的编号 | | | | |
| **参数说明** |  |  | | |  | |
| **返回值** |  | 返回当前登录用户的编号 | | | | |
| **范例** |  |  | | | | |
|  | | | | | |
| **服务名称** |  | **IExistenceService** | | | | |
| **说明** |  | **存在性检查服务，在数据库中检查是否存在数据** | | | | |
| **函数** |  | bool Exists(object oid) | | | | |
| **函数说明** |  | 根据Oid检查数据是否存在，适用于主实体 | | | | |
| **参数说明** |  | Oid | | | 主键值 | |
| **返回值** |  | 返回数据是否存在，存在返回true，否则返回false | | | | |
| **范例** |  | 判断单据类型是否存在    注意：Contxt后的‘DOC’一定要要是查询的目标typekey | | | | |
|  | | | | | | | |
| **函数** |  | bool Exists(string propertyName, object value) | | | | | | |
| **函数说明** |  | 根据属性名检查数据是否存在，适用于主实体 | | | | | | |
| **参数说明** |  | propertyName | | 属性名 | | | | |
|  | value | | 属性值 | | | | |
| **返回值** |  | 返回数据是否存在，存在返回true，否则返回false | | | | | | |
| **范例** |  | 判断销货单主表中的单号字段是否存在 | | | | | | |
|  | | | | | | | |
| **函数** |  | bool ExistsByPath(string path, object oid) | | | | | |
| **函数说明** |  | 根据Oid检查数据是否存在，根据Path不同查询主实体或明细实体 | | | | | |
| **参数说明** |  | path | 实体路径，例如SALES\_ORDER\_D | | | | |
|  | oid | Oid值 | | | | |
| **返回值** |  | 返回数据是否存在，存在返回true，否则返回false | | | | | |
| **案例** |  | 销货单中单身的多来源字段判断是否在销售单的子单身中是否存在。 | | | | | |

## 访问系统参数

E10中有参数表在数据库中，所有的参数都可以通过脚本直接访问到。

### 用户参数

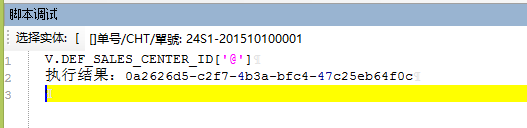
系统参数表都可以通过以下方式来访问：

**V.参数名 ['@']**

**范例：**

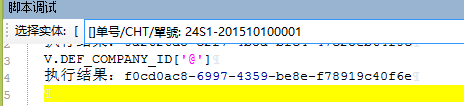
#### 获取用户的默认销售域id:

V.DEF\_SALES\_CENTER\_ID['@']



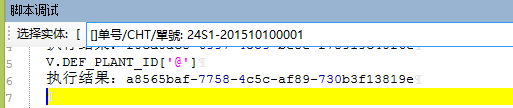
#### 获取用户的默认公司：

V.DEF\_COMPANY\_ID['@']



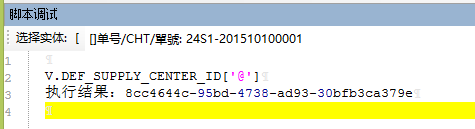
#### 获取用户的默认工厂

V.DEF\_PLANT\_ID['@']



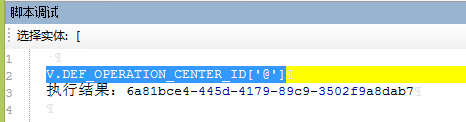
#### 获取用户的默认销售域服务工厂/储运id

V. DEF\_SUPPLY\_CENTER\_ID ['@']



#### 获取用户的默认营运域id

V.DEF\_OPERATION\_CENTER\_ID['@']



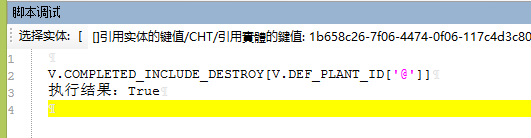
### 系统全局参数

系统参数表都可以通过以下方式来访问：

**V.参数名 [组织id]**

范例：需要访问工厂参数表（PARA\_PLANT）中的过账日依据字段（PLANT\_POST\_DATE\_BY）

V.COMPLETED\_INCLUDE\_DESTROY[V.DEF\_PLANT\_ID['@']]



只要知道参数所归属的组织，然后查询以下表，即可获取相关的系统参数

目前参数表在系统中是以下几个表：

|  |  |
| --- | --- |
| **表名** | **说明** |
| PARA\_BASESETTEMP | 门店模板参数 |
| PARA\_BC\_OP\_EMP | 工艺条码员工参数 |
| PARA\_COMPANY | 公司参数 |
| PARA\_DISTRIBUTION | 储运参数 |
| PARA\_GROUP | 公共参数 |
| PARA\_PLANT | 工厂参数 |
| PARA\_SALES\_CENTER | 销售域参数 |
| PARA\_POS | POS参数 |
| PARA\_OP\_TRACK | 生产追踪参数 |
| PARA\_SHOP | 门店参数 |
| PARA\_SUPPLY\_CENTER | 采购域参数 |