估值系统V3.0开发说明

Author: Paper Huang

Date: 2008-2-26

Last Update: 2014-07-23

**修改记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ver. No** | **发版日期** | **作者** | **审核人** | **改动的章节号** |
| 1.0 | 2008-2-26 | 黄芝平 |  | 初始版本 |
| 1.1 | 2008-5-7 | 黄芝平 |  | 修改：三.2.3.2.3公共业务模块  新增：三.2.6报表开发模板TreportListFrame |
| 1.2 | 2008-7-7 | 黄芝平 |  | 新增：三.4.1清算模块－数据导入配置  三.4.2报表的二次开发和配置  五.2清算模块－数据导入异常 |
| 1.3 | 2008-12-26 | 黄芝平 |  | 修改：三.4.1清算模块－数据导入配置  三.4.2报表的二次开发和配置  新增：四.1.2列表框ThsListBoxValue（Hundsun Lib3）  四.3主程序框架的Debug调试模式 |
| 1.4 | 2009-2-13 | 黄芝平 |  | 修改：三.2.3.2.2公共业务函数 |
| 1.5 | 2009-3-26 | 黄芝平 |  | 新增：三.2.3.2.4导出Excel和打印  四.1.7.多选下拉框hsCheckComboBox  四.2.3.日历选择下拉框ThsfDateTimePicker  删除：四.2.1.凭证控件TvoucherGrid（VoucherCtl） |
| 1.6 | 2009-12-28 | 黄芝平 |  | 新增：三.2.2.3.模块界面风格  三.2.3.3.框架功能函数  三.4.2.4.报表界面规范 |
| 1.7 | 2010-7-28 | 黄芝平 |  | 修改：二.3.第三方控件安装 |
| 1.8 | 2010-8-23 | 黄芝平 |  | 新增：二.5.操作系统设置 |
| 1.9 | 2011-07-28 | 黄芝平 |  | 新增：三.2.2.4.含业务列表导航栏的模块开发  四.4.多账套查询 |
| 2.0 | 2011-09-20 | 黄芝平 |  | 修改：二.3.第三方控件安装 |
| 2.1 | 2012-01-12 | 黄芝平 |  | 修改：三.2.2.3模块界面规范 |
| 2.2 | 2012-04-25 | 黄芝平 |  | 修改：三.2.2.3模块界面规范  三.4.2.4.报表界面规范  三.5.增值模块 |
| 2.3 | 2012-07-12 | 黄芝平 |  | 章节调整：  三.4.清算模块-数据导入配置  三.5.版本的二次开发和配置  三.6.增值模块  新增：  三.5.2.2.参数配置和取数方法  三.5.2.4.分组报表配置 |
| 2.4 | 2013-08-05 | 黄芝平 |  | 章节调整：  三.2.3.公共函数说明 |
| 2.5 | 2014-06-24 | 黄芝平 |  | 章节调整：  按照3.0部署要求重新整理第二章节 “环境部署”，增加中间件的部署说明和编译环境部署说明；  重新整理第三章的目录结构，按5个部分划分：程序框架、动态库开发、数据服务配置、数据导入配置、报表开发；  完善数据服务配置说明  完善数据导入配置说明 |

**目录**

[估值系统V3.0开发说明 1](#_Toc395011304)

[一. 关于本文档 6](#_Toc395011305)

[二. 环境部署 6](#_Toc395011306)

[1. 客户端运行环境部署 6](#_Toc395011307)

[2. 中间件运行环境部署 7](#_Toc395011308)

[2.1. Windows32运行环境 7](#_Toc395011309)

[2.1.1. 软件要求 7](#_Toc395011310)

[2.1.2. 配置授权 7](#_Toc395011311)

[2.1.3. 配置数据库连接 7](#_Toc395011312)

[2.1.4. 运行估值中间件 8](#_Toc395011313)

[2.2. Linux.i386运行环境 9](#_Toc395011314)

[2.2.1. OS要求 9](#_Toc395011315)

[2.2.2. 配置授权和环境变量 9](#_Toc395011316)

[2.2.3. 配置数据库连接 9](#_Toc395011317)

[2.2.4. 运行估值中间件 10](#_Toc395011318)

[3. 前台开发环境配置 10](#_Toc395011319)

[3.1. Delphi开发工具的版本 10](#_Toc395011320)

[3.2. Delphi开发环境Environment Options (建议配置) 10](#_Toc395011321)

[3.3. 第三方控件安装 11](#_Toc395011322)

[3.4. 配置控件库路径 12](#_Toc395011323)

[3.5. 操作系统设置 13](#_Toc395011324)

[3.6. 业务组件安装 13](#_Toc395011325)

[4. 中间件开发环境配置 14](#_Toc395011326)

[5. 前台编译环境配置 14](#_Toc395011327)

[5.1. Delphi开发工具的版本 14](#_Toc395011328)

[5.2. 编译基线版本和补丁版本 15](#_Toc395011329)

[5.2.1. 编译组件包 15](#_Toc395011330)

[5.2.2. 基线版 15](#_Toc395011331)

[5.2.3. 补丁版 15](#_Toc395011332)

[6. 中间件编译环境配置 16](#_Toc395011333)

[6.1. Windows32编译环境 16](#_Toc395011334)

[6.1.1. 软件要求 16](#_Toc395011335)

[6.1.2. 系统环境设置 16](#_Toc395011336)

[6.1.3. 编译数据服务插件 17](#_Toc395011337)

[6.1.4. 编译数据库插件 18](#_Toc395011338)

[6.2. Linux.i386编译环境 19](#_Toc395011339)

[6.2.1. OS要求 19](#_Toc395011340)

[6.2.2. 编译数据服务插件 19](#_Toc395011341)

[6.2.3. 编译数据库插件 19](#_Toc395011342)

[三. 开发说明 20](#_Toc395011343)

[1. 程序框架 20](#_Toc395011344)

[1.1. 功能特点 21](#_Toc395011345)

[1.2. 技术特点 22](#_Toc395011346)

[2. 动态库开发说明 22](#_Toc395011347)

[2.1. 动态库模块开发 22](#_Toc395011348)

[2.1.1. 工程文件选项 22](#_Toc395011349)

[2.1.1.1. 运行加载包Runtime package 23](#_Toc395011350)

[2.1.1.2. 版本信息Version Info 24](#_Toc395011351)

[2.1.1.3. 目录设置Directories 25](#_Toc395011352)

[2.1.2. 动态库开发模板 25](#_Toc395011353)

[2.1.3. 动态库模块入口函数 26](#_Toc395011354)

[2.1.4. 模块界面规范 27](#_Toc395011355)

[2.1.5. 含业务列表导航栏的模块开发 29](#_Toc395011356)

[2.2. 公用函数说明 29](#_Toc395011357)

[2.2.1. 技术类公共函数 29](#_Toc395011358)

[2.2.2. 业务类公共函数 30](#_Toc395011359)

[2.2.2.1. 数据服务类 30](#_Toc395011360)

[2.2.2.2. 公共业务函数 32](#_Toc395011361)

[2.2.2.3. 导出Excel和打印 35](#_Toc395011362)

[2.2.3. 框架功能函数 36](#_Toc395011363)

[2.3. 具备增删改特征模块的列表开发模板 38](#_Toc395011364)

[2.4. 模块整合和调试 42](#_Toc395011365)

[2.4.1. 菜单发布 42](#_Toc395011366)

[2.4.2. 动态库调试方法 42](#_Toc395011367)

[2.4.3. 使用数据服务的几种调试方法 42](#_Toc395011368)

[3. 数据服务 44](#_Toc395011369)

[3.1. 通用业务服务 44](#_Toc395011370)

[3.1.1. 调用接口 44](#_Toc395011371)

[3.1.2. 通用业务服务配置说明 45](#_Toc395011372)

[3.1.2.1. 【Query】数据查询 46](#_Toc395011373)

[3.1.2.2. 【Execute】执行脚本 49](#_Toc395011374)

[3.1.2.3. 【Procedure】执行过程 51](#_Toc395011375)

[3.1.2.4. 【GetField】字段取值 54](#_Toc395011376)

[3.1.2.5. 【UpdateField】更新字段 56](#_Toc395011377)

[3.1.2.6. 【ImpTable】数据表导入 58](#_Toc395011378)

[3.1.2.7. 【Log】日志记录 58](#_Toc395011379)

[3.1.2.8. 【Error】错误提示 59](#_Toc395011380)

[3.1.2.9. 【Method】调用方法 60](#_Toc395011381)

[3.1.2.10. 【Trans】执行事务 61](#_Toc395011382)

[3.1.2.11. 【Tree】树型转换 62](#_Toc395011383)

[3.2. 其他数据服务 62](#_Toc395011384)

[3.2.1. 调用接口 63](#_Toc395011385)

[3.2.1.1. Oracle数据库SQLLDR服务 63](#_Toc395011386)

[3.2.2. 其他数据服务开发说明（C++） 66](#_Toc395011387)

[4. 清算模块－数据导入配置 67](#_Toc395011388)

[4.1. TStoreType表结构说明 67](#_Toc395011389)

[4.2. 配置说明 69](#_Toc395011390)

[4.2.1. OLEDB接口方式导入ADO文件数据库 69](#_Toc395011391)

[4.2.1.1. 导入Excel文件 69](#_Toc395011392)

[4.2.1.2. 导入DBF文件 70](#_Toc395011393)

[4.2.2. SQLLoad导入格式化文本文件 72](#_Toc395011394)

[4.2.3. ADODB方式从外部数据库导入 75](#_Toc395011395)

[4.2.3.1. 数据源为Sql Server / DB2等 75](#_Toc395011396)

[4.2.3.2. 数据源为Oracle 76](#_Toc395011397)

[4.2.4. XML文件导入 77](#_Toc395011398)

[5. 报表开发 78](#_Toc395011399)

[5.1. 报表功能概述 79](#_Toc395011400)

[5.2. 报表开发工具－模板设计器 79](#_Toc395011401)

[5.2.1. 参数和函数表达式 79](#_Toc395011402)

[5.2.2. 配置参数和取数方法 80](#_Toc395011403)

[5.2.3. 设计报表『页面』 81](#_Toc395011404)

[5.2.4. 分组报表配置 82](#_Toc395011405)

[5.2.5. 交叉表配置 83](#_Toc395011406)

[5.3. 报表开发模板TReportListFrame 85](#_Toc395011407)

[5.4. 报表界面规范 85](#_Toc395011408)

[四. 其它说明 86](#_Toc395011409)

[1. 常用三方控件使用说明（推荐） 86](#_Toc395011410)

[1.1. 账套控件 86](#_Toc395011411)

[1.2. 交易对手 88](#_Toc395011412)

[1.3. 下拉框ThsComboBoxValue（Hundsun Lib） 88](#_Toc395011413)

[1.4. 列表框ThsListBoxValue（Hundsun Lib3） 89](#_Toc395011414)

[1.5. 带按钮的编辑框ComboEdit（RX Library） 89](#_Toc395011415)

[1.6. 带计算器的输入框TRxCalcEdit（RX Library） 89](#_Toc395011416)

[1.7. 数值和金额输入框TCurrencyEdit（RX Library） 89](#_Toc395011417)

[1.8. 表格控件TDBGridEh（EhLib） 89](#_Toc395011418)

[1.9. 多选下拉框hsCheckComboBox 90](#_Toc395011419)

[1.10. 日历选择下拉框ThsfDateTimePicker 91](#_Toc395011420)

[2. 特殊功能的三方控件使用 91](#_Toc395011421)

[~~2.1. 凭证控件TVoucherGrid（VoucherCtl）~~ 91](#_Toc395011422)

[2.2. 日历控件hsMonthCalendar 91](#_Toc395011423)

[3. 主程序框架的Debug调试模式 91](#_Toc395011424)

[4. 多账套查询（账套数大于1000） 92](#_Toc395011425)

[五. Q&A 92](#_Toc395011426)

[1. 数据集转换函数XmlToRecordSet异常 92](#_Toc395011427)

[2. 清算模块－数据导入异常 93](#_Toc395011428)

[3. 内存管理 94](#_Toc395011429)

[六. 附 95](#_Toc395011430)

[1. FA3.0中间件通讯协议 95](#_Toc395011431)

## 关于本文档

本文档力求整个产品组以统一的开发环境和配置来实现产品的稳定性、版本的统一。在编码方面，以统一的规范，保证编码方法和风格一致，提高后续维护性，降低后续维护和开发成本。

## 环境部署

### 客户端运行环境部署

客户端有两种工作模式：

直接连接数据库

连接FA中间件

两种工作模式是通过配置HundSun\_DB.xml区分的，需要根据实际使用模式切换：

【直接连接数据库】

|  |
| --- |
| <DBConnection ID="1" Caption="FA数据库" ConnectionString="Data Source=@Server;User ID=@User;Password=@Pass;PLSQLRSet=1;" SQLLOAD="@User/@Pass@@Server" SQLDAT="@Server/@User/@Pass">  <Provider DATATYPE="STRING">OraOLEDB.Oracle.1</Provider>  <ConnectionTimeout DATATYPE="INT">0</ConnectionTimeout>  <CommandTimeout DATATYPE="INT">0</CommandTimeout>  <Server Caption="数据库服务器" DATATYPE="STRING">orcl</Server>  <User Caption="登陆用户" DATATYPE="STRING">hsis0318</User>  <Pass Caption="用户口令" DATATYPE="STRING" Pass="\*">1F314DDB354B89B98447DBB5FDC64959</Pass>  </DBConnection> |

直连数据库时，需要运行HSClientCfg.exe程序配置数据库连接。

【连接FA中间件】

|  |
| --- |
| <T2Connection ID="1" Caption="FA服务器">  <ConnectionTimeout DATATYPE="INT">15</ConnectionTimeout>  <CommandTimeout DATATYPE="INT">30</CommandTimeout>  <Servers Caption="FA服务器" DATATYPE="STRING">127.0.0.1:9008</Servers>  <LicenseFile DATATYPE="STRING" Caption="许可证文件">(20100720)GTJAQH-HSUFT-0000\_C.dat</LicenseFile>  <SafeLevel>none</SafeLevel>  </T2Connection> |

### 中间件运行环境部署

目前可支持在Windows32、Linux.i386两种操作系统上运行。

#### Windows32运行环境

中间件的安装目录是instance

运行目录instance\workspace

##### 软件要求

需要已经安装了Oracle数据库

Oracle10g、Oracle9i 、Oracle11g（Oracle版本之一）

##### 配置授权

将获取到的授权文件放到instance\workspace\license目录下

检查instance\workspace\Data下面的DataService.xml及其子文件是否缺失

##### 配置数据库连接

打开配置文件instance\workspace\ls\_fa.xml，找到节点gz\_hsdb，如：

|  |
| --- |
| <plugin **lib**="**gz\_hsdb10**" getinfo="GetGZDBSvrInfo" load\_level="1" note="估值数据库访问">  <dss>  <ds logic\_name="1" conn\_count\_upper\_limit="25" wait\_conn\_timeout="10000" data\_source\_type="oracle" **server**="**orcl**" **user\_name**="**hsis0318**" **password**="**hsis0318**" database\_context="" init\_conn\_count="1" temp\_database="no" mark\_max\_busy\_count="no" mark\_all\_disconnect="no" init\_check="0"/>  </dss>  </plugin> |

修改加粗部分的内容：

lib --->> 数据库版本，gz\_hsdb9、gz\_hsdb10、gz\_hsdb11

server --->> 数据库服务名

user\_name --->> 用户名

password --->> 用户密码

运行config\_hsfa.bat加密数据库的用户密码

##### 运行估值中间件

打开文件夹instance\workspace

执行**start\_hsfa.bat**，正常启动有下列内容输出：

|  |
| --- |
| AppName:[ls\_fa] AppID:[0] SvrName:[mainsvr]  Plugin[com.hundsun.fbase.log] loaded(Ver=20090303,Sep 29 2013 17:13:46).  Plugin[com.hundsun.fbase.bizlog] loaded(Ver=20090303,Sep 29 2013 17:13:46).  Plugin[com.hundsun.fbase.config] loaded(Ver=20070704,Jun 30 2014 14:45:20).  Plugin[com.hundsun.fbase.f2packsvr] loaded(Ver=20101207,Mar 28 2014 14:29:21).  Plugin[com.hundsun.fbase.daemon] loaded(Ver=20080325,Sep 29 2013 17:13:41).  [7172]Plugin[com.hundsun.fbase.bizlog] init ...done!  [7172]Plugin[com.hundsun.fbase.config] init ...done!  [7172]Plugin[com.hundsun.fbase.daemon] init ...done!  [7172]Plugin[com.hundsun.fbase.f2packsvr] init ...done!  [7172]Plugin[com.hundsun.fbase.log] init ...done!  Plugin[t2] loaded(Ver=20090908,Mar 28 2014 14:29:31).  Plugin[com.hundsun.fbase.security] loaded(Ver=20071119,Sep 29 2013 17:13:48).  Plugin[mproxy] loaded(Ver=20080302,Sep 29 2013 17:13:53).  Plugin[com.hundsun.fbase.esbmessagefactory] loaded(Ver=20090108,Apr 4 2014 11:04:56).  Plugin[filter\_log] loaded(Ver=20090107,Sep 29 2013 17:14:45).  Plugin[router] loaded(Ver=20090526,Sep 29 2013 17:13:49).  Plugin[ospf] loaded(Ver=20081114,Dec 30 2013 15:15:48).  Plugin[f2] loaded(Ver=20081121,Sep 29 2013 17:14:52).  Plugin[proc] loaded(Ver=20081119,Mar 28 2014 14:29:33).  Plugin[com.hundsun.fbase.hsdb] loaded(Ver=20090309,Dec 30 2013 15:16:01).  Plugin[com.hundsun.fbase.gzdb] loaded(Ver=20090309,Jul 21 2014 15:12:31).  [7172]Plugin[com.hundsun.fbase.esbmessagefactory] init ...done!  [7172]Plugin[com.hundsun.fbase.security] init ...done!  [7172]Plugin[com.hundsun.fbase.gzdb] init ...done!  [7172]Plugin[com.hundsun.fbase.hsdb] init ...done!  [7172]Plugin[f2] init ...done!  [7172]Plugin[filter\_log] init ...done!  [7172]Plugin[mproxy] init ...done!  [7172]Plugin[router] init ...  \*\*\*\*\*\*\*\* Router Thread num is: 8\*\*\*\*\*\*\*\*\*\*\*\*\*  done!  [7172]Plugin[ospf] init ...done!  [7172]Plugin[proc] init ...done!  [7172]Plugin[t2] init ...T2:safelevel=[none] done!  Plugin[hsfa.dataservice] loaded(Ver=20140210,Jul 21 2014).  [7172]Plugin[hsfa.dataservice] init ...done!  [7172]ProcSysEvent[MSG\_START] begin ... T2:[IOCP] end!  [ls\_fa-mainsvr-0] STARTED |

#### Linux.i386运行环境

中间件的安装目录是/home/hsfa/instance

运行目录/home/hsfa/instance/workspace

##### OS要求

需要已经安装了Oracle数据库

Oracle10g、Oracle9i 、Oracle11g（Oracle版本之一）

##### 配置授权和环境变量

将获取到的授权文件放到/home/hsfa/instance/workspace/license目录下

检查/home/hsfa/instance/workspace/Data下面的DataService.xml及其子文件是否缺失

在当前用户根目录下，运行目录编译环境变量

编译环境变量$ vi .bash\_profile，增加下列内容

|  |
| --- |
| export LANG="zh\_CN.GB18030"  export NLS\_LANG="SIMPLIFIED CHINESE"\_CHINA.ZHS16GBK  export ORACLE\_HOME=/home/oracle/product/10.2.0  export FBASE\_HOME=/home/hsfa/instance/linux.i386  export LD\_LIBRARY\_PATH=$ORACLE\_HOME/lib:/lib:/usr/lib:/usr/local/lib:$FBASE\_HOME/lib  export PATH=$PATH:$HOME/bin:$FBASE\_HOME/bin:$ORACLE\_HOME/bin:$ORACLE\_HOME/lib:. |

##### 配置数据库连接

打开配置文件/home/hsfa/instance/workspace/ls\_fa.xml，找到节点gz\_hsdb，如：

|  |
| --- |
| <plugin lib="**gz\_hsdb10**" getinfo="GetGZDBSvrInfo" load\_level="1" note="估值数据库访问">  <dss>  <ds logic\_name="1" conn\_count\_upper\_limit="25" wait\_conn\_timeout="10000" data\_source\_type="oracle" server="**orcl**" user\_name="**hsis0318**" password="**hsis0318**" database\_context="" init\_conn\_count="1" temp\_database="no" mark\_max\_busy\_count="no" mark\_all\_disconnect="no" init\_check="0"/>  </dss>  </plugin> |

修改加粗部分的内容：

lib --->> 数据库版本，gz\_hsdb9、gz\_hsdb10、gz\_hsdb11

server --->> 数据库服务名

user\_name --->> 用户名

password --->> 用户密码

运行config\_hsfa.bat加密数据库的用户密码

##### 运行估值中间件

打开文件夹/home/hsfa/instance/workspace

执行start\_hsfa.sh，根据日志确定是否正常启动（参见Win32版本）

sh start\_hsfa.sh

### 前台开发环境配置

#### Delphi开发工具的版本

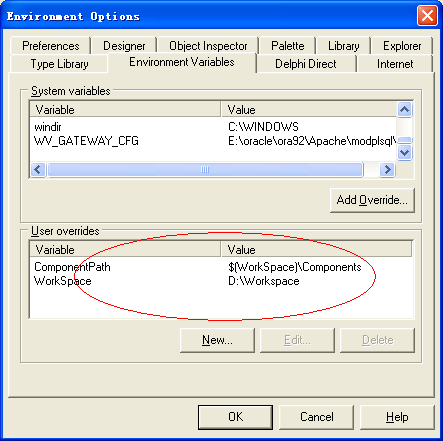


目前估值系统使用Delphi6＋Pack2进行开发。请注意在安装了Delphi6后打Pack2补丁。

Delphi开发工具要求安装英文版，不得安装中文版补丁和插件，不得安装编码格式化插件（格式化插件会自动修正代码格式，使其他人无法比对代码）。

#### Delphi开发环境Environment Options (建议配置)

建议配置好环境选项Environment Options中的环境变量，如下图：



在环境变量中增加两个变量：Workspace和ComponentPath，分别指向您的工作目录和控件目录。

有了环境变量，可以方便的相关资源的相对路径，避免团队开发中的配置路径不一致问题；也可以方便的只改环境变量值就切换到不同版本的工程代码。

#### 第三方控件安装

使用的三方控件列表

|  |  |  |
| --- | --- | --- |
| **控件** | **说明** | **使用模块** |
| EhLib | Eh表格控件，V2.1 | 财务、估值 |
| hsLib | Hundsun Lib | 财务、估值 |
| sncCurrency | 金额控件 | 凭证模块 |
| RX Library | RX控件 | 财务、估值 |
| Raize 4 | Raize控件 | 主程序、框架程序 |
| ExpressSpreadSheet | ExpressSpreadSheet | 报表模块 |
| Indy10 | Indy10的通信组件 | 自动更新模块、报表邮件模块 |
| AsyncPro | 传真服务组件 | 报表传真模块 |

\*\*注：标注的控件以后不再继续使用。

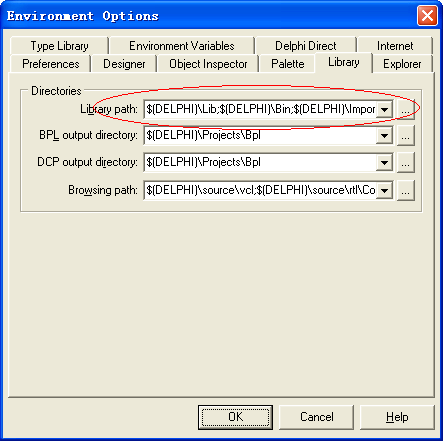
**红色**的控件必须安装，其他控件配置编译环境需要部署。

提示：开发过程中不得使用其它控件，如必须使用需上报申请。

控件的使用请查阅控件的帮助文档，或者本文后面的常用控件使用说明。

#### 配置控件库路径

将上述控件的安装路径配置到环境选项Environment Options中的Library Path中：



普通开发人员的建议配置为：

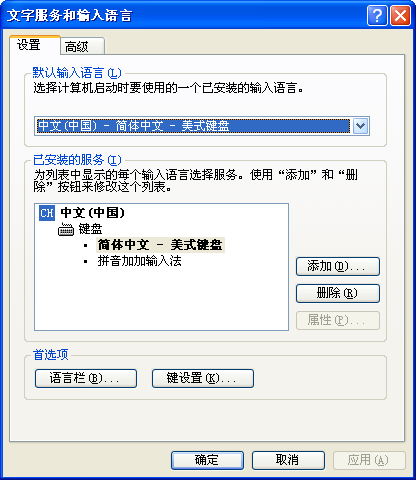
$(DELPHI)\Lib;$(DELPHI)\Bin;$(DELPHI)\Imports;$(DELPHI)\Projects\Bpl;$(ComponentPath)\Bpl;$(ComponentPath)\hsLib\Common;$(ComponentPath)\hsLib\Source;$(ComponentPath)\hsLib\delphi6;$(ComponentPath)\ehlib;$(ComponentPath)\rx275d6\Units;$(ComponentPath)\SpreadSheet1.25\XP Theme Manager\Sources;$(ComponentPath)\SpreadSheet1.25\ExpressGDI+ Library\Sources;$(ComponentPath)\SpreadSheet1.25\ExpressLibrary\Sources;$(ComponentPath)\SpreadSheet1.25\ExpressSpreadSheet\Sources;$(ComponentPath)\sncCurrency;$(ComponentPath)\Raize4\Source;$(ComponentPath)\Indy10\Lib\Protocols;$(ComponentPath)\Indy10\Lib\Core;$(ComponentPath)\Indy10\Lib\System

控件组开发人员可以根据实际需要在上述基础上增加其他控件。

#### 操作系统设置

【默认输入法】

默认输入法必须设置为“中文(中国) – 简体中文 – 美式键盘”。默认输入法设置为其它项，可能导致控件被设置成该输入法，编译出的程序也就被指定了输入法。



【环境变量】

检查操作系统的环境变量Path，包括用户变量和系统变量。Delphi6安装正常，Path中应该包含“Delphi6\Bin; Delphi6\Projects\Bpl;”两个目录的全路径。

如果安装了多个Delphi版本（如Delphi6＋Delphi7），则在使用Delphi6时，第三方控件容易出现问题。原因：不同版本的Delphi的bpl文件是不同名的，但第三方控件往往是同名的，通过path路径可查找到多个同名的不同版本的bpl文件。

#### 业务组件安装

使用的三方控件列表

|  |  |  |
| --- | --- | --- |
| **控件** | **说明** | **使用模块** |
| hsfaLib | Hundsun FA2.6 Lib  估值系统业务组件 | 财务、估值 |
| hsPrint | FA报表打印组件 | 财务、估值 |

### 中间件开发环境配置

【编码工具】

中间件采用标准的C/C++代码，不限定使用什么代码开发工具（如VS2010、CodeBlock等等）

【调试环境】

没有配置过调试环境，有兴趣的可尝试研究，将可行方法填补此处。

【编译Debug版本】

修改dev\src\utils\hsDebug.h文件，将第15行注释掉的代码打开：

#define \_DEBUG

重新执行编译命令，并运行程序。调试结果输出在Debug\Debug.log

### 前台编译环境配置

#### Delphi开发工具的版本



目前估值系统使用Delphi6＋Pack2进行开发。请注意在安装了Delphi6后打Pack2补丁。

Delphi开发工具要求安装英文版，不得安装中文版补丁和插件，不得安装编码格式化插件（格式化插件会自动修正代码格式，使其他人无法比对代码）。

#### 编译基线版本和补丁版本

##### 编译组件包

需要编译的三方组件列表

|  |  |  |
| --- | --- | --- |
| **控件** | **说明** | **使用模块** |
| EhLib | Eh表格控件，V2.1 | 财务、估值 |
| hsLib | Hundsun Lib | 财务、估值 |
| Raize 4 | Raize控件 | 主程序、框架程序 |

直接运行编译命令即可：MakeFile\_Bpl.cmd

##### 基线版

【全量编译】

MakeFileGroup.cmd

一般在新建基线版的第一次编译需要全量，发版的最后一次需要全量

【增量编译】

MakeFileGroup\_Inc.cmd

平常的定时编译任务使用增量编译

##### 补丁版

**注意：**新建一个补丁系列，需要从基线版编译后的Public目录拷贝下来；新建一个补丁，需要将上一个补丁的Public拷贝下来。

【全量编译】

MakeFileGroup.cmd

一般在新建补丁版的第一次编译需要全量，发版的最后一次需要全量

【增量编译】

MakeFileGroup\_Inc.cmd

平常的定时编译任务使用增量编译

### 中间件编译环境配置

目前需要编译Windows32、Linux.i386两个版本。

**注意：文件instance\workspace\license\hs\_unlimit.dat是内部使用的，不能打包到发布包中。**

#### Windows32编译环境

##### 软件要求

需要已经安装了下列软件

VC++6.0

Oracle10g、Oracle9i 、Oracle11g（Oracle版本之一）

##### 系统环境设置

【环境变量检查和设置】

安装VC++6.0以后，在用户环境变量下面，应该会自动生成以下3个环境变量

include 需要包含路径：C:\Program Files\Microsoft Visual Studio\VC98\include

lib 需要包含路径：C:\Program Files\Microsoft Visual Studio\VC98\lib

MSDevDir 需要包含路径：C:\Program Files\Microsoft Visual Studio\Common\MSDev98

在用户环境变量或者系统环境变量Path

Path 需要包含路径：C:\Program Files\Microsoft Visual Studio\VC98\Bin

【测试VC++可用性】

1、开始🡪运行🡪cmd🡪输入cl

查看输出，如果有VC++的版本则证明已经安装成功。

2、将如下代码保存至d:\helloworld.cpp

|  |
| --- |
| #include<stdio.h>  int main()  {  printf("Hello World!\n");  return 0;  } |

CMD下输入： cl d:\helloworld.cpp

应该显示如下列内容：

|  |
| --- |
| D:\>cl helloworld.cpp  Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8168 for 80x86  Copyright (C) Microsoft Corp 1984-1998. All rights reserved.  helloworld.cpp  Microsoft (R) Incremental Linker Version 6.00.8168  Copyright (C) Microsoft Corp 1992-1998. All rights reserved.  /out:helloworld.exe  helloworld.obj  D:\>helloworld.exe  Hello World! |

##### 编译数据服务插件

执行dev\src\make.bat，编译成功有下列内容输出：

|  |
| --- |
| Microsoft (R) Incremental Linker Version 6.00.8168  Copyright (C) Microsoft Corp 1992-1998. All rights reserved.  /out:service\_impl.dll  /dll  /implib:..\Lib\win32\hsfa\_dataservice.lib  /out:..\Lib\win32\hsfa\_dataservice.dll  ..\Lib\win32\f\_os.lib  /EXPORT:GetDATASERVICEInfo  /EXPORT:GetDATASERVICEInstance  /EXPORT:DATASERVICEInit  /EXPORT:GetDATASERVICEDependSvr  service\_impl.obj  StatusControl.obj  static\_hash.obj  hsMemStream.obj  XMLFactory.obj  calculator.obj  hsStrUtils.obj  hsDebug.obj  recordsettoxml.obj  hsDataSet.obj  DBOperator.obj  hsProcParameterMap.obj  hsDataService\_cfg.obj  fsc\_dataservice\_impl.obj  hsDataService\_Class.obj  hsFileService.obj  hsDS\_Content\_Sql.obj  hsDS\_Content\_Method.obj  hsDS\_Content\_Procedure.obj  hsDS\_Content\_Field.obj  hsDS\_Content\_Log.obj  hsDS\_Content\_ImpTable.obj  hsDS\_Content\_Trans.obj  hsFS\_SqlLdr.obj  Creating library ..\Lib\win32\hsfa\_dataservice.lib and object ..\Lib\win32\hsfa\_dataservice.exp  F:\Workspace\HsFA2.6\Sources\Hsfa3.0\_FD\dev\src>del \*.obj  F:\Workspace\HsFA2.6\Sources\Hsfa3.0\_FD\dev\src>copy ..\Lib\win32\hsfa\_dataservice.dll ..\..\inst  ance\win32\lib  已复制 1 个文件。 |

如果包含Warn、Error，需要程序开发人员修改优化代码。

##### 编译数据库插件

该插件的编译需要依赖于OCI库，需要在编译文件中加入当前的数据安装环境信息。

打开文件夹dev\src\gzdb

1）修改make.bat，根据需要编译的Oracle版本，确定调用makefile9、makefile10、makefile11，并修改指定的动态库文件gz\_hsdb9.dll、gz\_hsdb10.dll、gz\_hsdb11.dll

nmake -f makefile10.mvc all

copy ..\..\Lib\win32\gz\_hsdb10.dll ..\..\..\instance\win32\lib

2）配置编译参数

查看makefile10.mvc（根据Oracle版本确定makefile9、makefile10、makefile11）

配置#Oracle安装目录：

ORACLE\_HOME = d:\oracle\product\10.2.0\db\_1

注：需要根据oralce版本及路径设置环境变量，此例是安装了Oracle 10g.2

3）执行make.bat，编译成功有下列内容输出：

|  |
| --- |
| fsc\_hsgzdb\_impl.obj  gz\_connection\_impl.obj  gz\_data\_source\_impl.obj  gz\_statement\_impl.obj  gz\_result\_set\_impl.obj  gz\_datetime.obj  gz\_ora.obj  gz\_callable\_statement\_impl.obj  hs\_string.obj  Creating library ..\..\Lib\win32/gz\_hsdb.lib and object ..\..\Lib\win32/gz\_hs  db.exp  D:\SVN\FD-3.0\dev\src\gzdb>del \*.obj  D:\SVN\FD-3.0\dev\src\gzdb>copy ..\..\Lib\win32\gz\_hsdb.dll ..\..\..\instance\wi  n32\lib  已复制 1 个文件。 |

#### Linux.i386编译环境

##### OS要求

服务器已经安装 gcc编译器，gcc版本要求：

g++ 3.4.6 （或者4.1.2，版本越低越好）

Oracle10g、Oracle9i 、Oracle11g（Oracle版本之一）

目前在虚拟机部署的编译环境的版本如下

Linux版本：Red Hat Enterprise Linux Server release 5.5 (Tikanga)

gcc版本：4.1.2 20080704 (Red Hat 4.1.2-48)

##### 编译数据服务插件

打开文件夹../dev/src

执行编译指令$ sh make.sh

分析编译信息warn、error，联系开发人员优化修改

##### 编译数据库插件

该插件的编译需要依赖于OCI库，需要在编译文件中加入当前的数据安装环境信息。

打开文件夹../dev/src/gzdb

1）修改make.sh，根据需要编译的Oracle版本，确定调用makefile9、makefile10、makefile11，并修改指定的动态库文件gz\_hsdb9.so、gz\_hsdb10.so、gz\_hsdb11.so

make -f makefile10.gcc

cp ../../Lib/linux.i386/libgz\_hsdb10.so ../../../instance/linux.i386/lib/

2）配置编译参数

查看makefile10.gcc（根据Oracle版本确定makefile9、makefile10、makefile11）

编辑文件vi makefile10.gcc

配置#Oracle安装目录：

ORACLE\_HOME = /home/oracle/product/10.2.0

注：需要根据oralce版本及路径设置环境变量，此例是安装了Oracle 10g.2

3）执行编译指令$ sh make.sh，

分析编译信息warn、error，联系开发人员优化修改

## 开发说明

估值系统V2.6、V3.0版本采用动态库模块化开发，本章节描述动态库开发的相关事项。

### 程序框架

估值主程序HSValuate.exe是新版系统的程序框架，支持了V2.6版本的动态库模块接口，附属了数据库连接模块、数据服务引擎、页面引擎、登录模块等。

业务模块以动态库形式开发，主程序中不包含业务处理。

【程序结构图】



如上图，三层模式时，服务端程序运行在中间件上。两层模式时，服务端程序运行在本地客户端。

#### 功能特点

1. 支持多个子系统，每个系统登录后完全独立的菜单和桌面，系统界面类XP风格；
2. 支持多文档窗口和模式窗口；主窗口有左侧参数栏，支持多文档窗口将参数栏统一置入其中；
3. 主程序包含主页窗口，其中有O32的功能菜单树，系统桌面功能（可用户自定义快捷方式）；
4. 权限部分各子系统统一管理，为用户－用户组模式，即用户可以多继承用户组的模块权限。模块权限支持可定义的功能划分（如可以将菜单配置模块的功能定义为：浏览、修改、配置）。
5. 数据连接模块支持各种数据库连接，并可以同时配置多个数据库连接（Ole驱动和参数可以在配置文件中配置）；
6. 数据服务引擎采用服务导向思路设计，支持以服务接口（XML接口）的方式配置各种业务逻辑；

统一处理了数据库连接的申请、连接异常处理、连接断开重连等问题；

替代了DataModule模块的原先作用，代码更简洁；

客户端编码和SQL脚本完全分离，便于代码检查审核和切换数据库；

提高编码调试效率，**AutoRefresh**和**Debug**开关给编码带来更多的方便性；

数据库防火墙作用，提高数据库安全性；

本地内存表作用，数据敏感控件修改数据不再直接操作数据库，程序逻辑更简单；

1. 除权限模块涉及财务账套外，框架的其它模块与业务无关联，可以应用到其它业务系统中。

#### 技术特点

1. 动态库模块化，框架包括主程序、登录校验模块、系统管理模块、数据库连接模块、数据服务引擎模块、配置窗体模块；
2. 框架可以对各模块窗体进行编码补充，如设置窗体的字体和字符集、多文档窗体关闭后的自动释放；
3. 两次和三层模式可切换，配置连接模式模块即可；
4. 数据服务引擎支持即时修改调试，有调试开关，配置数据业务逻辑时比编码更有优势（方便灵活、可配置大部分业务逻辑）；引擎的数据处理部分采用自编码的SAX\_XML进行数据结构解析和构建；引擎的处理结果可以处理为ADORecordset，该数据集独立于数据库而存在与内存中，可以作内存表使用；
5. 登录校验模块在登录和修改口令时采用DES3进行口令加密，可独立升级；

### 动态库开发说明

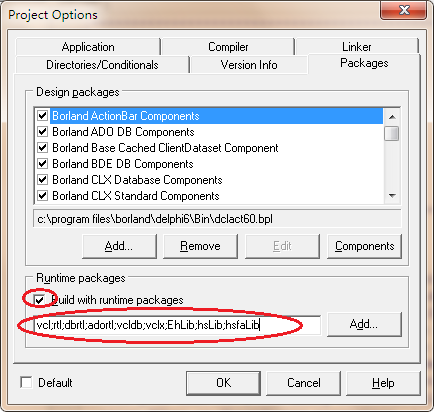
#### 动态库模块开发

##### 工程文件选项

★ 在DllModel目录中有动态库工程文件模板，打开DllModel.dpr另存为您所需要的工程文件即可。动态库模板中已包括本节中描述的工程文件选项设置（版本信息的文件描述需要重新设置）。

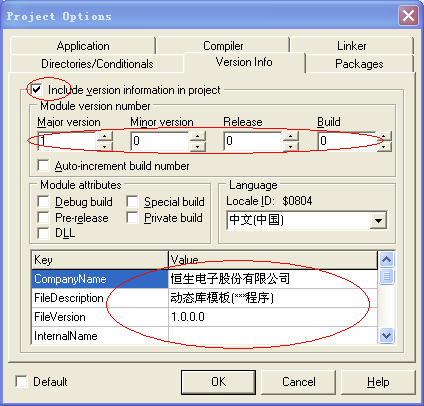
用Delphi打开工程文件，在工程选项Project Options中，需要设置运行加载包Runtime packages、版本信息Version Info、目录设置Directories

###### 运行加载包Runtime package



由于系统的动态库模块化开发需要，须在菜单Project Options的Packages页中Build with runtime packages打勾并输入“vcl;rtl;dbrtl;adortl;vcldb;vclx;EhLib;hsLib;hsFaLib”

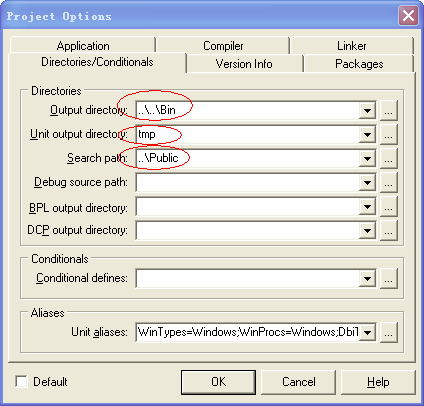
###### 版本信息Version Info



下面列出需要设置的版本信息内容：

|  |  |  |
| --- | --- | --- |
| **Key** | **Value** | **说明** |
| CompanyName | 恒生电子股份有限公司 | 公司名称 |
| FileDescription | ××程序 | 应用程序说明，请注意改写 |
| FileVersion | 1.0.0.0 | 文件版本 |
| ProductName | 资产估值与会计核算系统 | 产品名称 |
| ProductVersion | V2.6 | 产品版本，配置管理员发布时统一设置 |

###### 目录设置Directories



|  |  |  |
| --- | --- | --- |
| **Key** | **Value** | **说明** |
| Output directory | ..\..\Bin | 工程编译输出的目标路径 |
| Unit output directory | tmp | 编译连接文件输出路径，新建工程时需要在工程目录下新建tmp目录 |
| Search path | ..\Public | 公用函数路径 |

##### 动态库开发模板

新建动态库时推荐使用动态库开发模板。新建一个动态库模块时需要设置前面所述的『工程文件选项』，通过动态库模块可以省略大部分操作。

【动态库开发模板特征】

运行加载包设置（已完成）

版本信息设置（部分完成，需要修改文件描述）

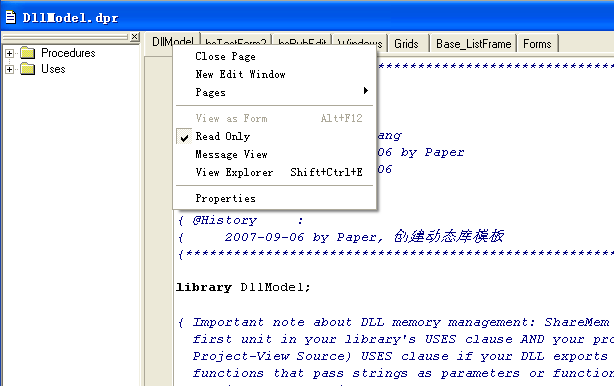
目录设置（已完成）

运行参数Run Parameters（已完成）：用于调试运行

模块功能发布函数说明和示例（需要根据实际功能改写）

【动态库开发模板使用方法】

用Delphi打开DllModel目录下的DllModel.dpr，打开后在代码页上打开右键菜单（如下图所示），确认不为Read Only（如果是，点击一次即为否），然后在新目录下另存为指定的新模块名称。



然后设置工程文件的版本信息（修改文件描述FileDescription）、定义发布接口函数和接口说明。

##### 动态库模块入口函数

动态库模块的调用是通过统一接口的不同入口函数方法调用的，可以在相同或者不同的动态库中开发实现。

【模块入口函数原型】

TEntryForm =

function(MethodID: WideString; const sParam: WideString; out sError: WideString): TForm; stdcall;

MethodID：方法ID，可以忽略不用

sParam：方法参数，可以忽略不用

sError：错误信息输出，可以忽略不用

Function Result：返回构建的模块窗体对象（创建对象即可，由框架负责显示、释放、模块进入日志等）。

【入口函数示例】

|  |
| --- |
| //测试模块入口  function TestForm(MethodID: WideString; const sParam: WideString;  out sError: WideString): TForm; stdcall;  begin  Result := TForm.Create(Application);  end; exports TestForm; |

【入口函数的关键要素】

参数个数、参数类型

**Stdcall**

##### 模块界面规范

1、模块主界面的工具栏和参数栏；



对应的右键菜单和快捷键



注意：鼠标双击时默认为查看，BaseListFrame框架里面已经封装了该功能。

2、**弹出式编辑窗口**

从TBaseDialogForm继承，窗口居中Position=MainFormCenter，

BorderType=bsDialog, KeyPressView=true

编辑控件的垂直间距为32像素（每32像素放置一行，控件高度20像素，间隔12像素）

标签左对齐，必输项在右侧加“＊”，且在递交前需要调用CheckValid来校验输入是否为空。

账套控件：宽度200像素

编辑控件的标准宽度：200像素

界面大小，两列排版时界面标准宽度640，单列排版时界面标准宽度400

小于等于10行的按单列排版，否则按照两列排版。

在窗口的设计界面中右键菜单“FA Dialog”下面有两个子菜单“Reset all(IS)”、“Reset all(FD)”，执行后可对界面自动排版。对Visible属性为false的组件不参与排版。

界面宽度500以内时，按单列排版，500以往按两列排版。

查看和复核界面调用AllReadOnly方法实现所有输入为只读。查看仅显示“取消”按钮，复核界面显示“复核”、“取消”

编辑界面可通过ControlReadOnly方法设置指定的输入框为只读，确定时调用CheckValid来校验输入是否为空，校验不合法时定位焦点。

编辑窗口按钮为“确定”和“取消”、“取消”、“保存”和“关闭”三组之一。

选择“保存”和“关闭”时，保存按钮不能关闭窗口，关闭时如果未保存需要提示保存。

“取消”按钮需要有Esc快捷键。

3、**标签：**界面标签不带中文或者英文的冒号

标签文本修改（有特殊含义的除外）：

“账套选择：”统一改成“**选择账套**”

“时间”、“选定日期”等统一改成“**选择日期**”，日期时间段统一改为“2008-11-11～2008-11-12”的风格（两个日期控件之间加中文符号“～”）。

标签位于输入框左侧，同列的所有标签左对齐，并与输入框居中对齐，标签文本中不加中英文的冒号。

4、**表格**：

**标题居中**，文本**左对齐**、数值**右对齐**，上下内容长度一致的允许居中（如日期）。

有隔行色，右键菜单，新建、删除、查看或者编辑的快捷键。

金额显示格式“#,###.00”，万元显示格式“#,###.000000”

数量显示格式“#,###”，“#,###.######”。整数和小数并存时数量的小数位为最大6位动态显示。

表格列数很多的（需要滚动显示列的），需要将关键列前置并设置冻结列

上下复核模式时，未复核记录的字体颜色hsPubUtils.clFont\_UnVerify

列宽度：30/80/100/120/200

一般含表格的业务界面都要求嵌入BaseListFrame（除报表、财务、系统框架模块外）。该页面框架包括了以下常用功能（详见开发样板DllModel.dll/ hsListFrame2.pas）：

定义列功能GetColEh、SetColEh

多选功能MultiSelect，是否允许选择的事件OnCanSelect

5、**下拉框**：下拉个数设置为16/24/32，且必须默认选中一项。

6、**消息框**

消息框按钮为“确定”、“确定”“取消”、“是”“否”“取消”三组之一。

“确定”提示信息

“确定”“取消”为一般的询问消息框，缺省焦点在取消按钮上。

“是”“否”“取消”为保存提示消息框，缺省焦点在取消按钮上，选择取消时不关闭窗口或者不切换导航数据。

消息框提示范例：

1. 确认删除
   1. 删除账套将删除关联的所有业务数据和设置！请确认是否删除账套“\*\*”？[确定 取消]
   2. 请确认是否删除所选的债券买卖数据？[确定 取消]
2. 确认保存
   1. \*\*已经被修改，请确认是否保存？[是(Y) 否(N) 取消]
3. 提示
   1. 请输入『债券代码』 [确定]

7、**字体**

模块窗体和弹出对话框的字体为：宋体9号字，字符集为GB2312。

8、合理设置Tab键顺序：从左到右、从上到下。

Enter键和Tab键都要能够跳转。

##### 含业务列表导航栏的模块开发

动态库模板（Detail\DllModel\DllModel.dpr ）中带有导航栏的界面开发模板。

1、业务列表导航栏风格的开发模板为hsTestForm2.pas，可将该单元另存为新的文件

    模块主界面中包括导航栏（树形）、工具栏、参数栏：

      。导航栏切换各业务界面并刷新数据、切换工具栏按钮与业务界面的关联

      。参数栏参数变更触发当前业务界面的数据刷新

      。业务界面刷新时，需要主界面提供参数xml

      。如需要功能权限细分的控制，在主界面处理

2、每个业务界面继承TBaseListFrame （todo: new -> other -> 对应工程文件名称 -> BaseListFrame）

    业务界面实现数据刷新、新增、修改、删除、复核等操作

**注意：**业务界面不要新建frame，而是继承TBaseListFrame

#### 公用函数说明

##### 技术类公共函数

【位置】Control/Common/\*.pas

|  |  |  |
| --- | --- | --- |
| **单元** | **函数**（**红色**为常用函数） | **说明** |
| hsWinUtils  Win函数 | **hsMsgBox** | 消息对话框 |
| **hsInputQuery/hsInputBox** | 文本输入对话框 |
| **LocalPath** | 应用程序所在的本地路径 |
|  |  |
| hsDateUtils  时间函数 | **hsDateToStr** | 日期转换为字符串（格式:yyyymmdd） |
| **hsTimeToStr** | 时间转换为字符串（格式:hh:nn:ss） |
| **hsDateTimeToStr** | 时间转换为字符串（格式:yyyymmdd hh:nn:ss） |
| **hsStrToDateTime** | 时间字符串转换为时间，支持3种格式（yyyymmdd or yyyymmdd hh:nn:ss or hh:nn:ss） |
|  |  |
| hsStrUtils  字符串处理 | **hsStrReplace** | 查找替换字符串, 功能可选项(全替换、忽略大小写、全单词匹配) |
| StrPosRight/AnsiPosRight | 从右侧开始查找字符串 |
| StrToArray | 将字符串按特定字符分隔, 输出String数组 |
|  |  |
| hsFileUtils  文件操作 | （略）  详见单元函数说明 |  |
| hsIniUtils  注册表和配置文件操作 | （略）  详见单元函数说明 | 注册表键值判断是否存在  注册表键值增、删、读、写  配置文件Ini键值增、删、读、写  全局变量IniFilename已经初始化为应用程序名称（后缀名改为.ini） |
| hsNetUtils  网络操作 | （略）  详见单元函数说明 |  |

★注意：时间和字符串的转换函数必须使用hsDateUtils中的对应方法，禁用Delphi自带的DateToStr和StrToDate方法。Delphi自带的DateToStr和StrToDate方法和本地机器的区域设置有关。

##### 业务类公共函数

###### 数据服务类

【位置】Public/DllInterface.pas

**数据库连接**

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **GetDBConnection**

description : 获取数据库连接, 并对未建立的连接参试连接

parameter : dbID 数据库ID, 一般传入空字符串

return : DBConn 已建立连接,返回ADO连接对象; 连接失败, 返回空

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function GetDBConnection(dbID: WideString): TADOConnection;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

procedure : **RestoreDBConnection**

create : 2008-12-08, by Paper

description : 回退数据库连接，将连接归还数据库连接池

parameter : DBConn 数据库ADO连接对象

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

procedure RestoreDBConnection(DBConn: TADOConnection);

**数据服务接口**

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **BussinessMethod**

description : 通用业务服务接口函数

parameter : MethodID 业务方法ID

parameter : sParam 业务传入参数,一般为xml格式

parameter : sData 业务返回结果参数,一般为xml格式

parameter : sError 业务返回错误提示信息

return : 返回值>=0表示成功, 返回值>0的更多业务含义需参照对应业务接口定义

返回值<0表示错误代码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function BussinessMethod(MethodID: WideString; const sParam: WideString;

out sData: OleVariant; out sError: WideString): Integer;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **RecordSetToXml**

create : 2007-09-27, by Paper

description : ADO数据集转换为XML文本格式

parameter : RS 传入的ADO数据集

parameter : sData 返回的XML文本格式

return : 返回值表示转换成功的数据集个数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function RecordSetToXml(RS: ADOInt.Recordset; out sData: OleVariant): Integer;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **XmlToRecordSet**

create : 2007-09-27, by Paper

description : XML文本转换为数据集

parameter : sXml 传入的XML文本的数据集

parameter : RS 返回ADO结果数据集

return : 返回值表示转换成功的数据集个数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function XmlToRecordSet(sXml: PChar; out RS: ADOInt.Recordset): Integer;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **XmlToRecordSetM**

create : 2008-09-02, by Paper

description : XML文本转换为数据集列表

parameter : sXml 传入的XML文本的数据集

parameter : RS 返回ADO结果数据集列表

return : 返回值表示转换成功的数据集个数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function XmlToRecordSetM(sXml: PChar; out RS: TRecordSetList): Integer;

###### 公共业务函数

【位置】Public/**hsPubUtils.pas**

**获取数据字典**

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **FillDictionaryToItems**

create : 2008-02-28, by Paper

description : 根据字典类别名称填充字典条目

parameter : DictName传入的字典类别名称，或者：

（"证券类别", "交易市场", "用户账套", "管理账套", "基金账套",

"结算币种", "用户列表", "用户代码"）

parameter : Keys 传入的TStrings列表对象，用于存放返回的字典条目编码

parameter : Items 传入的TStrings列表对象，用于存放返回的字典条目名称

return : 返回值表示条目个数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function FillDictionaryToItems(DictName: String; Keys, Items: TStrings): Integer;

示例1：下拉框（下拉框使用市场编号，显示市场名称）

|  |
| --- |
| //初始化交易市场。boxMarket为Hundsun Lib3控件，请点击连接查看：[ThsComboBoxValue](#_下列框ThsComboBoxValue（Hundsun_Lib3）)  if FillDictionaryToItems('交易市场', boxMarket.Values, boxMarket.Items)>0 then  boxMarket.ItemIndex := 0; |

示例2：表格列值显示为名称（市场编号显示为市场名称）

|  |
| --- |
| //初始化表格列，使表格TDBGridEh的市场编号列，显示为市场名称。控件请点击连接查看：[TDBGridEh](#_表格控件TDBGridEh（EhLib）)  procedure TAccountMngForm.FormCreate(Sender: TObject);  var  colEh: TColumnEh;  begin  colEh := GetColEh('l\_sclb');  if Assigned(colEh) then  FillDictionaryToItems('交易市场', colEh.KeyList, colEh.PickList);  end; |

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **FillDictionaryToItems\_All**

create : 2008-02-28, by Paper

description : 根据字典类别名称填充字典条目

如果条目数量≥2在首行增加值为"#"名称为"<全部>",

parameter : DictName 传入的字典类别名称，或者：

（"证券类别", "交易市场", "用户账套", "管理账套", "基金账套",

"结算币种", "用户列表", "用户代码"）

parameter : Keys 传入的TStrings列表对象，用于存放返回的字典条目编码

parameter : Items 传入的TStrings列表对象，用于存放返回的字典条目名称

return : 返回值表示条目个数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function FillDictionaryToItems\_All(DictName: String; Keys, Items: TStrings): Integer;

**获取系统参数**

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **GetSysParam**

create : 2008-11-18, by Paper Huang

description : 获取系统参数

parameter : FundID 账套编号

sParam 参数代码

sDefault 参数缺省值

return : 返回参数值

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function GetSysParam(const FundID: Integer;

sParamCode, sDefault: String): String; overload;

function GetSysParam(const FundID: Integer;

sParamCode, sDefault: String; out sParam: String): Integer; overload;

示例1：获取系统参数（根据账套获取估值表模板的文件名称）

|  |
| --- |
| if GetSysParam(FundID, 'GZBBMB', '证券投资基金估值表.xcr', sFile) >= 0 then  begin  sFile := ChangeFileExt(sFile, '.xcr');  Rpt.ModuleFileName := GetReportDir +'\' + sFile;  end; |

示例2：获取系统参数（新股处理时判断是否核算投资类型）

|  |
| --- |
| if GetSysParam(strtoint(cbZT.Value),'JYQS\_SFHSTZLX','2') = '2' then //不核算  begin  cbTZLX.Enabled := false;  cbTZLX.ItemIndex := -1;  cbTZLX.Color := clInactiveCaptionText;  end  else  begin  cbTZLX.Enabled := true;  cbTZLX.Color := clWindow;  if cbTZLX.Value = '' then  cbTZLX.Value := '2';  end; |

**发送系统消息**

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **PopupSysMessage**

create : 2009-01-23, by Paper Huang

description : 向系统消息处理模块发送提示消息

parameter : sParam 消息包体XML

<Message Type="Message, Warning, Error">

<Title>消息标题(摘要)</Title>

<Detail>详细信息(文本)</Detail>

</Message>

return : 返回参数值

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function PopupSysMessage(const sParam: WideString): Integer;

**判断增值模块**

界面要求判断是否开启增值模块，未开启的，相应界面要屏蔽相关功能。

指定的增值模块是否开启的函数

**function** **GetAddedValue**(vID: String): Boolean;

**function** GetAddedValue(vID: Integer): Boolean;

**当前系统时间和记账日期**

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **GetSysDate**

create : 2009-03-03, by liuwj

description : 获取当前系统时间（数据库时间）

parameter :

return : 当前系统时间(带时分秒)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function GetSysDate: TDatetime;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **GetAccountDate**

create : 2008-03-17, by Lizq

description : 获取记账日期

return : 返回值记账日期

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function GetAccountDate: TDatetime;

**选择证券信息对话框**

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **SelectStock\_A**, **SelectStock\_B**, **SelectStock\_C**

create : 2008-03-31, by Paper Huang

description : 选择证券公用对话框

parameter : L\_ZQNM 返回的证券内码

V\_ZQMC 返回的证券名称

AssetType 证券类别, 输入参数, 对SelectStock\_C函数为输入输出参数

MarketType 市场编号, 输入参数,对SelectStock\_C函数为输入输出参数

return : 返回值, 是否确认选择

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function SelectStock\_A(var L\_ZQNM: Integer;

AssetType: String=''; MarketType: String=''): Boolean;

function SelectStock\_B(var L\_ZQNM: Integer; var V\_Code, V\_ZQMC: String;

AssetType: String=''; MarketType: String=''): Boolean;

function SelectStock\_C(var L\_ZQNM: Integer; var V\_Code, V\_ZQMC: String;

var AssetType: String; var MarketType: String): Boolean;

**选择科目对话框**

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **SelectAccount**

create : 2009-08-03, by liuwj

description : 选择科目公用对话框

parameter : Code 科目代码 输入输出参数，输入值用于定位用

CodeName 返回科目代码加名称 例：10020101\_活期存款

LeafOnly 是否必须选择叶子节点 true:是 false:否

MultiSelected 多选科目 true:多选 false:单选

return : 返回值, 是否确认选择

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function SelectAccount(var Code,CodeName: string; FundID: Integer;

LeafOnly: Boolean=true; MultiSelected: Boolean=false): Boolean;

###### 导出Excel和打印

【位置】Public/**hsExportUtils.pas**

该单元包含了导出Excel和打印两个action，当ReportView属性指向有文本标签、输入框、日历、DBGrid、DBGridEh等组件，或者指向包含有这些组件的父面板时，可以将这些组件全部导出或者打印。

示例1：导出excel设置

|  |
| --- |
| //创建Excel导出的action  acExp\_Excel := TRepExportExcel.Create(self);  acExp\_Excel.Name := 'actExportExcel';  acExp\_Excel.Caption := '导出Excel';  acExp\_Excel.Hint := '导出到Excel';  acExp\_Excel.ImageIndex := 25; //Excel按钮的图标  acExp\_Excel.OnUpdate := self.acSelectUpdate;  **TRepExportExcel**(acExp\_Excel).ReportView := RptViewPanel; //指向需要导出的数据组件或其父面板  TRepExportExcel(acExp\_Excel).SaveFileDlg := true;  NacExcel.Action := acExp\_Excel;//关联右键菜单  btnExcel.Action := acExp\_Excel;//关联按钮  btnExcel.Caption := '导出'; |

示例2：打印设置

|  |
| --- |
| //创建打印的action  acExp\_Print := TRepPrintAction.Create(self);  acExp\_Print.Name := 'actReportPrint';  acExp\_Print.Caption := '打印';  acExp\_Print.Hint := '打印';  acExp\_Print.ImageIndex := 0; //打印按钮的图标  acExp\_Print.OnUpdate := self.acSelectUpdate;  TRepExportExcel(acExp\_Print).ReportView := RptViewPanel; //指向需要打印的数据组件或其父面板  btnPrint.Action := acExp\_Print;//关联右键菜单  NacPrint.Action := acExp\_Print;//关联按钮 |

##### 框架功能函数

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CurrentUserID : 当前用户ID

CurrentUserCode : 当前用户编码

CurrentUserName : 当前用户名称

CurrentSubSysID : 当前子系统ID

CurrentAccountBook 当前账套

CurrentAccountState 当前登陆的账套状态(1:业务账套, 2:运作中, 3:已结束, 4:统计账套)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function CurrentUserID : Integer;

function CurrentUserCode: String;

function CurrentUserName: String;

function CurrentSubSysID: String;

function CurrentAccountBook: String;

function CurrentAccountState: String;

示例1：

|  |
| --- |
| procedure TTestForm.FormCreate(Sender: TObject);  begin  FillDictionaryToItems('用户账套', hsComboBoxValue1.Values, hsComboBoxValue1.Items);  hsComboBoxValue1.Value := **CurrentAccountBook**;  end; |

注：**尽量避免使用ReadIniValue**(IniFilename, 'SysInfo', 'FundID', s)

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**账套切换后各模块的账套自动切换功能**

建议在各业务数据录入模块使用自动更新，结果查询的模块不需要。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

示例1：

//刷新账套, 在窗口对象上声明一个接收消息的函数

procedure HsMsgProc(var Message: TMessage); message WMU\_HS\_Msg;

procedure TTestForm.HsMsgProc(var Message: TMessage);

begin

case Message.WParam of

HS\_Msg\_AccBook:

hsComboBoxValue1.Value := **CurrentAccountBook**;

//LParam 也是更新后的账套ID, 也可以这样使用:

//hsComboBoxValue1.Value := IntToStr(Message.LParam);

end;

end;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **GetFormGrantFunc**

create : 2008-8-25, by Paper

description : 根据窗体句柄获取公共赋权模块（非账套赋权）的当前用户权限, 不能在窗体构造函数中调用, 在构造后调用

parameter : AFormHandle 当前窗体句柄

return : 返回值=0表示该模块未定义功能赋权, 或者对当前用户只读权限

返回值>0表示对当前用户赋权, 每位二进制值表示一项对应权限

返回值<0表示错误代码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function GetFormGrantFunc(AFormHandle: THandle): Integer;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **GetFormModuleInfo**

create : 2010-4-7, by Paper

description : 根据窗体句柄获取模块信息, 不能在窗体构造函数中调用, 在构造后调用

parameter : AFormHandle 当前窗体句柄

parameter : SubSysID 返回值，模块所属的子系统ID

parameter : MID 返回值，模块ID

return : 返回值, 是否取到模块信息

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function GetFormModuleInfo(AFormHandle: THandle;

out SubSysID, MID: String): Boolean;

#### 具备增删改特征模块的列表开发模板

很多模块界面都具备增删改和列表显示特征，即模块窗口显示内容列表，通过增删改菜单或者按钮增删改内容。使用列表开发模板可以节省很多界面的布局、状态控制、快捷键等各种费时又容易忽略的工作，程序可读性、可维护性和稳定性也有提高。

【列表开发模板特征】

列表表格有统一的隔行颜色、选择颜色

有统一的增删改、刷新、查找菜单，通过关联TAction实现状态控制和执行

有统一的菜单图标和快捷键（增、删、刷新）

关闭当前模块窗口acClose: TAction

【列表开发模板位置】

Public\**Base\_ListFrame.pas**

【列表开发模板使用方法：界面排版】

1. 新建窗口，设置窗体的属性使符合规范

字体：宋体，9号，GB2312

边框：BorderStyle=bsSizeable

位置：Position=poMainFormCenter

标题：模块名称

显示按钮提示：ShowHint=true

1. 在窗口中加入工具栏TToolBar，设置属性：

向上靠齐（Align=alTop）

显示下边框（EdgeBorders=[ebBottom]）

显示按钮标题（ShowCaptions=true）

高度37像素（Height=37）

扁平效果（Flat=true）

1. 确认工程中已经加入了列表模板单元，在窗口中加入列表模板框架（控件栏选择Standard→Frames，在窗口空白处点击，在弹出的Frame选择框中选择模板框架BaseListFrame）
2. 设置列表框架对象靠齐方式为alClient，在工具栏中依次加入工具按钮并设置Action为列表框架的（acAdd/ acEdit/ acDel/ acRefresh），再加入退出按钮请编写退出代码（Self.Close）。
3. 如果有参数栏，则一般为TPanel：

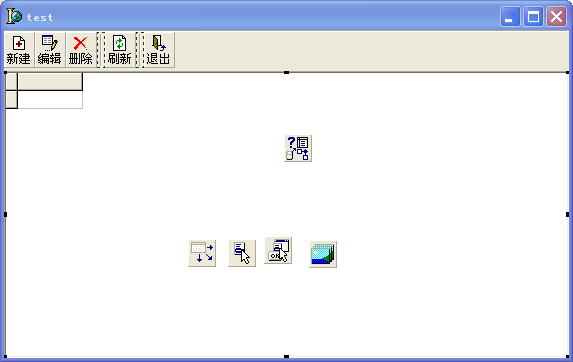
扁平效果，无边框，工具栏高度37像素

每个参数的格式为 “标签＋参数框”，标签不带冒号，参数一般为单行且上下居中，第一个参数标签居左8个像素

选择账套   下拉框，宽度为161，下拉个数16或者更多（24、32）

选择日期   日期框，宽度为105，Format='yyyy-MM-dd'

界面排版效果图：



【列表开发模板的取数和增删改接口】

接口方法：

property LookField: String; //主键（关键字段，新增、刷新后重新定位用。多个字段用分号“;”分割。未设置时缺省为第一个字段）

property OnRefresh: TRefreshEvent; //刷新事件（重新取出列表数据）

property OnAddNode: TAddNodeEvent; //新建事件

property OnEditNode: TEditNodeEvent; //编辑事件

property OnDelNode: TEditNodeEvent; //删除事件

接口方法原型：

TRefreshEvent = procedure(var ADataSet: TDataSet) of object;

TAddNodeEvent = function(var NodeID: Variant): Boolean of object;

TEditNodeEvent = function(const NodeID: Variant): Boolean of object;

【接口说明】

1. 刷新事件（取数接口）

procedure(var ADataSet: TDataSet)

参数ADataSet，传递方向（in、out），为当前列表框架对应的数据集。

在该接口中实现取数方法。

示例1：

|  |
| --- |
| procedure TTestForm.DoDataRefresh(var ADataSet: TDataSet);  begin  with ADOQuery1 do  begin  Connection := DllInterface.GetDBConnection('');  SQL.Text := 'select t.\* from trights t';  Open;  ADataSet := ADOQuery1;  end;  end; |

示例2：（推荐方法，使用到的相关函数参见[公共函数说明](#_公用函数说明)）

|  |
| --- |
| procedure TAccountMngForm.DoDataRefresh(var ADataSet: TDataSet);  var  sData: OleVariant;  sError: WideString;  rs: \_Recordset;  begin  //获取账套列表，业务编码：Account\_List  if BussinessMethod('Account\_List', '', sData, sError)>=0 then  begin  XmlToRecordSet(PChar(String(sData)), rs);  if Assigned(rs) then  begin  FDataSet.Recordset := rs;  ADataSet := FDataSet;  end;  end  else if sError<>'' then  hsMsgBox(sError, '获取账套列表', MB\_ICONWARNING+MB\_OK);  end; |

1. 新建事件（新增数据接口）

function(var NodeID: Variant): Boolean;

参数NodeID为新建的业务对象ID，或者为标识该新建业务对象的唯一索引，即与LookField对应的字段值。

该接口中实现新建，建议在新的编辑窗口操作，确认后将新建的业务对象ID返回到NodeID，用于重新刷新定位到新建的记录上。

示例1：（下面列举LookField为多个字段的处理方法）

|  |
| --- |
| function TTestForm.DoDataAdd(var NodeID: Variant): Boolean;  var  sData: OleVariant;  sError: WideString;  begin  Result := false;  With TTestItemDefForm.create(nil) do  Try  If ShowModal=mrOk then  Begin  NodeID := VarArrayCreate([0, 2], varVariant);  NodeID[0] := VarFromDateTime(dtCmd.Date);  NodeID[1] := trim(edtYBBM.value);  NodeID[2] := trim(edtBBBM.value);  //新增数据的递交处理  ……  Result := True;  //处理结束  End;  Finally  Free;  End;  end; |

1. 修改事件（修改数据接口）

function(const NodeID: Variant): Boolean;

参数NodeID为修改的业务对象ID，即与LookField对应的字段值。

如果LookField字段数为一个，NodeID为该字段的值；如果LookField字段数为多个，NodeID为变体类型的数组：variant array= VarArrayCreate([0, n-1], varVariant)

该接口中实现修改，建议在新的编辑窗口操作。

1. 删除事件（删除数据接口）

function(const NodeID: Variant): Boolean;

删除事件的参数同修改事件。

★注意：删除前，必须提示用户进行删除确认，消息框的焦点在取消按钮上，以免快速操作时误删。

示例1：

|  |
| --- |
| function TAccountMngForm.DoDataDel(const NodeID: Variant): Boolean;  var  sData: OleVariant;  sError: WideString;  begin  Result := false;  if MessageBox(PChar('请确认是否删除账套定义项“'+FDataSet.FindField('vc\_name').AsString+'”？'),  '确认删除', MB\_ICONQUESTION + MB\_OKCANCEL + MB\_DEFBUTTON2) = mrOk then  begin  Result := BussinessMethod('Account\_Del',  Format('<Param AccountID="%d" />', [Integer(NodeID)]), sData, sError)>0;  end;  end; |

#### 模块整合和调试

##### 菜单发布

1）菜单的模块配置

在TSys\_Module表配置模块，其中入口函数字段VC\_URL配置格式为“动态库名称|入口函数名称”，或者为“ ，如：

hsAccountCfg.dll|AccountMngForm

hsDictionary.dll|ShowDictionaryConfig

hsReport.dll|ShowReportForm|资产负债表

CustForm.dll|CustEntryForm|Set\_QSKJZ

注意入口函数名称是区分大小写的，要与动态库exports函数名称完全一致。

2）赋予用户的模块期限，重新登录估值系统。

##### 动态库调试方法

建立ProjectGroup，加入主程序和相关的动态库工程文件，可以方便调试动态库。

##### 使用数据服务的几种调试方法

使用数据服务引擎可以方便开发，极大的提高编码调试效率，**AutoRefresh**和**Debug**开关给编码带来更多的方便性【注意：只在两次框架下支持调试模式】。

开启方法：在配置文件头设置开关为true。

|  |
| --- |
| <CustomService AutoRefresh="**true**" Debug="**true**"> |

【开启实时更新开关**AutoRefresh**】

在数据服务配置文件更新保存后，下次运行即按修改后的配置运行，不需要重新登录系统。

【开启调试开关**Debug**】

配置的数据库脚本执行错误时，只要客户端按照编码规范将错误信息显示，就可以在消息框中显示错误的接口方法ID、数据库错误提示和Sql脚本。

【错误调试示例】

客户端规范编码：

|  |
| --- |
| if BussinessMethod('Account\_List', '', sData, sError)>=0 then  begin  ……  end  else if sError<>'' then  hsMsgBox(sError, '获取账套列表', MB\_ICONWARNING+MB\_OK); |

假设脚本语法错误，消息框提示：



当前活动窗体是该消息框，按快捷键『Ctrl+C』可以将消息框复制为文本到剪切板，粘贴到文本编辑器显示为：

|  |
| --- |
| ---------------------------  获取账套列表  ---------------------------  MethodID=Account\_List  Error=-2147217900  ORA-00904: "A"."L\_VOID1": 无效的标识符  select a.\* from tsysinfo a where '' is null or a.L\_VOID1=''  ---------------------------  确定  --------------------------- |

### 数据服务

#### 通用业务服务

通用数据服务以给定业务编号（MethodID: String）访问，或者给定服务配置（sConfig: String，xml格式）访问。

##### 调用接口

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **BussinessMethod**

create : 2007-09-27, by Paper

description : 通用业务服务接口函数

parameter : MethodID 业务方法ID

parameter : sParam 业务传入参数,一般为xml格式

parameter : sData 业务返回结果参数,一般为xml格式

parameter : sError 业务返回错误提示信息

return : 返回值>=0表示成功, 返回值>0的更多业务含义需参照对应业务接口定义

返回值<0表示错误代码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function BussinessMethod(MethodID: WideString; const sParam: WideString;

out sData: OleVariant; out sError: WideString): Integer; overload;

function BussinessMethod(MethodID: String; const sParam: String;

out sData: OleVariant; out sError: WideString): Integer; overload;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **BussinessMethodEx**

create : 2007-09-27, by Paper

description : 通用业务服务接口函数

parameter : sConfig 业务方法配置xml格式

parameter : sParam 业务传入参数,一般为xml格式

parameter : sData 业务返回结果参数,一般为xml格式

parameter : sError 业务返回错误提示信息

return : 返回值>=0表示成功, 返回值>0的更多业务含义需参照对应业务接口定义

返回值<0表示错误代码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function BussinessMethodEx(sConfig: WideString; const sParam: WideString;

out sData: OleVariant; out sError: WideString): Integer; overload;

function BussinessMethodEx(sConfig: String; const sParam: String;

out sData: OleVariant; out sError: WideString): Integer; overload;

##### 通用业务服务配置说明

【通用配置说明】

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| 根节点 | 模块自动化接口的根节点 | 1属性AutoRefresh：布尔值，缺省为false。为true时程序能够根据更新时间自动刷新配置，但响应速度较慢。项目交付前请将该刷新重置为false。在两层模式下有效。  2属性Debug：布尔值，缺省为false。为true时，程序执行数据库脚本错误后，在返回的错误信息中包含脚本信息。 |
| Package | 数据服务接口包 | 接口包，起分类归档的作用。  1属性Desc：接口的描述，便于维护和记录日志  2属性Verify：校验信息，便于开发维护记录 |
| Proc | 一个自动化数据服务接口的配置节点 | 1属性**ID**：接口编码，对应于服务程序或者客户端程序的调用接口。如果对应于客户端接口，该属性必须是数字型，并根据整体分配唯一编号。  2属性Return：接口函数的返回值重置定义，缺省不设置该属性。定义该属性后，接口返回值将根据定义的整形参数值返回。  3属性Desc：接口的描述，便于维护和记录日志  4属性Verify：校验信息，便于开发维护记录 |
| Param | 自动化接口的参数定义。接口自动化实现时将根据该定义从XML请求参数中提取执行参数。 | 1属性**Code**：参数编码，在该接口中必须是唯一编码。  2属性Value：参数值  3属性ValuePath：属性Value不存在时有效，从XML请求参数中提取值的XPath方法。（用于Sql脚本中的带“@”前缀符的参数值替换；注意这里是对Sql语句的字符串替换，需要关注替换后的语法）  4属性Type：参数值类型，缺省为空，目前用于定义大字段类型，在执行内容为参数法时有效。  参数值类型，目前用于定义大字段类型，在执行内容为参数法时有效。缺省为空，即对普通的参数替换或者参数值  Blob:Text 将文本转存到Blob参数或者字段  Blob:Hex 将已经转换为16进制文本还原为二进制数据转存到Blob参数或者字段  Blob:File 将文件转存到Blob参数或者字段 |
|  | 系统默认参数。接口自动化实现时将当前连接的用户ID、用户代码、子系统代码作为参数。 | 1默认参数**UserID**，当前连接的用户ID  2默认参数**UserCode**，当前连接的用户编码  3默认参数**SysCode**，当前连接的子系统代码  4默认参数**Return**，当非首个执行内容时，该参数表示前一个内容的执行结果值 |
| Content | 接口实现的每个步骤，在这里称为执行内容。  接口的实现由一个或者多个连续的执行内容构成。  注：右侧格式说明为各种执行内容都具备的格式描述，各种类型的特有格式描述详见后面说明。 | 1位置：Proc的子节点，可以有多个内容节点。  2属性DBID：当前内容执行的数据库连接ID，对应于数据库连接配置文件HundSun\_DB.xml中的连接ID；缺省为配置中的第一个连接  3属性**Type**：执行内容的类型，目前支持11种类型：数据查询Query、数据执行Execute、存储过程Procedure、数据字段取值GetField、数据字段更新UpdateField、批量更新UpdateBatch、生成树结果Tree、方法接口调用Method、错误Error、事务处理Trans、表记录修改日志Log  4属性Condition：内容执行的条件，为布尔型表达式，表达式中引用的参数用“@参数编码”表示，支持数值运算和字符串逻辑判断。  例如Condition="@temp\_c=1"。  5属性LoopPath：内容循环执行的XMLPath条件，其中各参数的ValuePath的方法中也必须增加循环序号[#Index]表示参数提取的位置。  例如LoopPath="Item"。  6属性Finally：该属性标记为强制执行项，不论前面执行是否正确。 |
| Content/ Param | 临时参数节点，一般从查询结果或者方法接口调用结果中提取  Content Type="Query"  Content Type="Method" | 1位置：Content的子节点，可以有多个参数节点。  2属性**Code**：参数编码，在该接口中必须是唯一编码。如果重复则会导致取值被覆盖  3属性ValuePath：从结果XML中提取值的XPath方法。例如从字段名称为‘X’的结果中提取，属性ValuePath配置为： ValuePath="METADATA/ROWDATA/ROW/@X"（从查询结果或者方法调用的返回结果中取值的参数）  4属性Field：从结果集RecordSet取字段值的方法，配置的值为字段名称，不区分大小写。 |

###### 【Query】数据查询

**配置说明**

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Content | 接口实现的每个步骤，在这里称为执行内容。  接口的实现由一个或者多个连续的执行内容构成。 | 1属性DBID：当前内容执行的数据库连接ID，对应于数据库连接配置文件HundSun\_DB.xml中的连接ID；缺省为配置中的第一个连接  2属性**Type**：数据查询Query  3属性Condition：内容执行的条件，为布尔型表达式，表达式中引用的参数用“@参数编码”表示，支持数值运算和字符串逻辑判断。  例如Condition="@temp\_c=1"。  [详情见上文“公共配置”]  4属性LoopPath：内容循环执行的XMLPath条件，其中各参数的ValuePath的方法中也必须增加循环序号[#Index]表示参数提取的位置。  例如LoopPath="Item"。  5节点值text()：执行内容的表达式（字符串或者脚本等），格式为数据库操作的脚本 |
| Content/ Param | 临时参数节点，一般从查询结果或者方法接口调用结果中提取 | 1属性**Code**：参数编码，在该接口中必须是唯一编码。  参数值  2属性ValuePath：从结果XML中提取值的Xpath方法。例如从字段名称为‘X’的结果中提取，属性ValuePath配置为： ValuePath=”METADATA/ROWDATA/ROW/@X”（注②） |

注① 内部取值参数，在Sql语法中定义为以“:”的参数，在存储过程中为输入或者输入输出型参数；注意这里**不是**参数替换，而是作为Sql语法或者存储过程中的参数取值。

内容Content定义参数参数代码.ValuePath，该参数用于该内容的内部提取参数，优先获取参数；如果未定义参数代码.ValuePath，则首先从内部临时参数取值；未找到临时参数，缺省从XML请求参数中的根节点属性取值（属性代码为参数代码）。

IS20130306之前的版本用OraEleDB的驱动，在处理Insert、Select、Update、Delete等查询、执行类的脚本时，不支持参数取值[由于驱动本身不能刷新参数个数和类型]；但可支持对存储过程、函数的参数取值法。

IS20130306及以后的版本，已经完全支持在Insert、Select、Update、Delete、Procedure等的参数取值法。

注② 从查询结果或者方法调用的返回结果中取值的参数。

**示例1**

【Query配置】

|  |
| --- |
| <Proc ID="DicClass\_List" Desc="根据分类获取字典条目列表">  <Param Code="DicClassFlag" ValuePath="@DicClassFlag"/>  <Content Type="Query">  Select a.c\_lemma\_item,a.vc\_item\_name, a.l\_dictionary\_no  From tdictionary a  Where a.L\_DICTIONARY\_NO='@DicClassFlag' and a.C\_LEMMA\_ITEM!='!'  by a.c\_lemma\_item  </Content>  </Proc> |

【调用参数sParam格式】

|  |
| --- |
| <Param DicClassNo=”10201”/> |

【客户端调用】

|  |
| --- |
| **procedure** TDictionaryForm.ClassDoDataRefresh(var ADataSet: TDataSet);  **var**  sData: OleVariant;  sError: WideString;  rs: \_Recordset;  **begin**  //DicClass\_List  if BussinessMethod('DicClass\_List',  Format('<Param DicClassNo="%s" />',  [vartostr(FCatDataSet.FieldValues['L\_DICTIONARY\_NO'])]),  sData, sError)>=0 then  begin  XmlToRecordSet(PChar(String(sData)), rs);  if Assigned(rs) then  begin  FClassDataSet.Recordset := rs;  ADataSet := FClassDataSet;  end;  end  else  if sError<>'' then  hsMsgBox(sError, '获取字典条目', MB\_ICONWARNING+MB\_OK);  **end**; |

**示例2**

【用例说明】

查询结果中获取参数，并返回参数值

【Query配置】

|  |
| --- |
| <Proc ID="CheckTableExist" Desc="查找目的表是否存在" Return="findcount">  <Param Code="TBName" ValuePath="@TBName"/>  <Param Code="TBType" ValuePath="@TBType"/>  <Content Type="Query">  SELECT Count(1) C FROM TAB  WHERE TNAME=UPPER('@TBName')  AND (('@TBType' is null and TABTYPE='TABLE') or TABTYPE=UPPER('@TBType'))  <Param Code="findcount" ValuePath="METADATA/ROWDATA/ROW/@C"/>  </Content>  </Proc> |

【调用参数sParam格式】

|  |
| --- |
| <Param TBName=”Show2003” TBType=”TABLE”/> |

【客户端调用】

|  |
| --- |
| **function** CheckTableExist(ATable: String; out sError: WideString;  TBType: String='TABLE'): Boolean;  **var**  sData: OleVariant;  iResult: Integer;  **begin**  iResult := BussinessMethod('CheckTableExist',  Format('<Param TBName="%s" TBType="%s"/>',  [UpperCase(ATable), UpperCase(TBType)]), sData, sError);  Result := iResult > 0;  **end**; |

**示例3**

【用例说明】

查询指定非估值数据源

【Query配置】

|  |
| --- |
| <Proc ID="GetDBUserList" Desc="获取数据库中用户列表">  <Content Type="Query" DBID="DBUser">  select t.vc\_sys,t.vc\_user,'\*\*\*\*\*\*' vc\_psw,t.d\_alter, t.L\_CRYPT  from tsys\_dbuser t  where upper(t.vc\_sys) &lt;&gt; 'DBUSER'  order by t.d\_alter desc,t.vc\_sys  </Content>  </Proc> |

###### 【Execute】执行脚本

**配置说明**

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Content | 接口实现的每个步骤，在这里称为执行内容。  接口的实现由一个或者多个连续的执行内容构成。 | 1属性DBID：当前内容执行的数据库连接ID，对应于数据库连接配置文件HundSun\_DB.xml中的连接ID；缺省为配置中的第一个连接  2属性**Type**：执行数据库脚本Execute  3属性Condition：内容执行的条件，为布尔型表达式，表达式中引用的参数用“@参数编码”表示，支持数值运算和字符串逻辑判断。  例如Condition="@temp\_c=1"。  [详情见上文“公共配置”]  4属性LoopPath：内容循环执行的XMLPath条件，其中各参数的ValuePath的方法中也必须增加循环序号[#Index]表示参数提取的位置。  例如LoopPath="Item"。  5节点值text()：执行内容的表达式（字符串或者脚本等），格式为数据库操作的脚本 |

注① 内部取值参数，在Sql语法中定义为以“:”的参数，在存储过程中为输入或者输入输出型参数；注意这里**不是**参数替换，而是作为Sql语法或者存储过程中的参数取值。

内容Content定义参数参数代码.ValuePath，该参数用于该内容的内部提取参数，优先获取参数；如果未定义参数代码.ValuePath，则首先从内部临时参数取值；未找到临时参数，缺省从XML请求参数中的根节点属性取值（属性代码为参数代码）。

IS20130306之前的版本用OraEleDB的驱动，在处理Insert、Select、Update、Delete等查询、执行类的脚本时，不支持参数取值[由于驱动本身不能刷新参数个数和类型]；但可支持对存储过程、函数的参数取值法。

IS20130306及以后的版本，已经完全支持在Insert、Select、Update、Delete、Procedure等的参数取值法。

**示例1**

【用例说明】

给定字典类型和字典条目编号，删除该条目

【Execute配置】

|  |
| --- |
| <Proc ID="DicClass\_Del" Desc="删除字典条目">  <Param Code="CatID" ValuePath="@CatID"/>  <Param Code="ClassID" ValuePath="@ClassID"/>  <Content Type="Execute">delete from tdictionaryL\_DICTIONARY\_NO=@CatID and C\_LEMMA\_ITEM='@ClassID'</Content>  </Proc> |

【调用参数sParam格式】

|  |
| --- |
| <Param CatID="10201" ClassID="8"/> |

【客户端调用】

|  |
| --- |
| function TDictionaryForm.DoDataDel(const NodeID: Variant): Boolean;  var  sData: OleVariant;  sError: WideString;  begin  Result := false;  if MessageBox(PChar('请确认是否删除字典条目 “'+FDataSet.FindField('vc\_name').AsString+'”？'),  '确认删除', MB\_ICONQUESTION + MB\_OKCANCEL + MB\_DEFBUTTON2) = mrOk then  begin  Result := BussinessMethod(DicClass\_Del',  Format('<Param CatID="%d" ClassID="%s"/>', [Integer(NodeID[0]), vartostr(NodeID[1])]), sData, sError)>0;  end;  end; |

**示例2**

【用例说明】

保存用户组（角色）的模块权限，先删除所有权限，然后批量插入权限

【Execute配置】

|  |
| --- |
| <Proc ID="SetGroupRight\_Module" Desc="保存指定用户组的模块权限">  <Param Code="GroupID" ValuePath="@GroupID"/>  <Content Type="Execute"><![CDATA[  delete from TSys\_UserGroup\_Right  where L\_GID=@GroupID  ]]></Content>  <Param Code="SysCode" ValuePath="Module[#Index]/@SysCode"/>  <Param Code="MDCode" ValuePath="Module[#Index]/@MDCode"/>  <Param Code="FuncID" ValuePath="Module[#Index]/@FuncID"/>  <Content Type="Execute" LoopPath="Module"><![CDATA[  insert into TSys\_UserGroup\_Right(L\_GID, Vc\_Sys, Vc\_Code, l\_Funcid)  values (@GroupID, '@SysCode', '@MDCode', @FuncID)  ]]></Content>  </Proc> |

【调用参数sParam格式】

|  |
| --- |
| <Param GroupID="1">  <Module SysCode="Valuate" MDCode="OptionTradeForm" FuncID="3"/>  <Module SysCode="Valuate" MDCode="MetalTradeForm" FuncID="5"/>  <Module SysCode="Valuate" MDCode="ETFArbitrage" FuncID="7"/>  </Param> |

###### 【Procedure】执行过程

**配置说明**

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Content | 接口实现的每个步骤，在这里称为执行内容。  接口的实现由一个或者多个连续的执行内容构成。 | 1属性DBID：当前内容执行的数据库连接ID，对应于数据库连接配置文件HundSun\_DB.xml中的连接ID；缺省为配置中的第一个连接  2属性**Type**：执行存储过程Procedure  3属性Condition：内容执行的条件，为布尔型表达式，表达式中引用的参数用“@参数编码”表示，支持数值运算和字符串逻辑判断。  例如Condition="@temp\_c=1"。  [详情见上文“公共配置”]  4属性LoopPath：内容循环执行的XMLPath条件，其中各参数的ValuePath的方法中也必须增加循环序号[#Index]表示参数提取的位置。  例如LoopPath="Item"。  5属性**ProcName**：表示存储过程的名称  属性Return：存储过程返回参数中，表示错误代码的参数名称  6属性Error：存储过程返回参数中，表示错误信息的参数名称  7属性Params：存储过程传入的参数，使用“=”作为参数名和参数值的分隔符，使用“；”作为不同参数之间的分隔符，可能需要做参数替换。  8存储过程返回参数不需要定义，除定义到属性Return和属性Error中的返回值，其它的返回参数都可以作为下次内容执行的参数。 |

注① 内部取值参数，在Sql语法中定义为以“:”的参数，在存储过程中为输入或者输入输出型参数；注意这里**不是**参数替换，而是作为Sql语法或者存储过程中的参数取值。

内容Content定义参数参数代码.ValuePath，该参数用于该内容的内部提取参数，优先获取参数；如果未定义参数代码.ValuePath，则首先从内部临时参数取值；未找到临时参数，缺省从XML请求参数中的根节点属性取值（属性代码为参数代码）。

IS20130306之前的版本用OraEleDB的驱动，在处理Insert、Select、Update、Delete等查询、执行类的脚本时，不支持参数取值[由于驱动本身不能刷新参数个数和类型]；但可支持对存储过程、函数的参数取值法。

IS20130306及以后的版本，已经完全支持在Insert、Select、Update、Delete、Procedure等的参数取值法。

**示例1**

【用例说明】

调用估值表的存储过程，查询或者生成估值表

【Procedure配置】

|  |
| --- |
| <Proc ID="Rpt\_Guzibiao\_Jijin" Desc="证券投资基金估值表">  <Param Code="FundID" ValuePath="@FundID"/>  <Param Code="DDate" ValuePath="@Date"/>  <Param Code="Audit" ValuePath="@Audit" Desc="审核标记：0生成数据后查询, 1确认审核, -1取消审核, 2不重新生成数据直接查询"/>  <Content Type="Query">select user||'.SP\_RPT\_GZB' PROCNAME from dual  <Param Code="ProcName" ValuePath="METADATA/ROWDATA/ROW/@PROCNAME"/>  </Content>  <Content Type="Procedure" ProcName="@ProcName" Params="ztbh=@FundID;RQ=@DDate;aAudit=@Audit;a\_vc\_checker=@UserCode" Return="return\_code" Error="return\_str"/>  </Proc> |

【客户端调用】

返回的数据包是**多结果集**的，需要调用XmlToRecordSetM转换为结果集数组

|  |
| --- |
| if BussinessMethod('Rpt\_Guzibiao\_Jijin',  Format('<Param FundID="%d" Date="%d" Audit="%d"/>',  [MyFundInfo.FundID, Trunc(MyFundInfo.QueryDate), Audit]),  sData, sError) >= 0 then  begin  XmlToRecordSetM(PChar(String(sData)), rsList);  ……  end; |

**示例2**

【用例说明】

调用表达式的函数，返回函数执行结果

【Procedure配置】

|  |
| --- |
| <Proc>  <Content Type="Procedure" ProcName="sf\_pub\_kmye" Params="a\_l\_ztbh=1012;a\_vc\_code=1002;a\_l\_sl=0;a\_d\_currdate=40011"/>  </Proc> |

返回**XML**格式的结果：

|  |
| --- |
| <Data datatype ="number">1.012</Proc> |
| <Data datatype ="string">Hello Word</Proc> |
| <Data datatype ="datetime">2014-8-5 14:05:22</Proc> |
| <Data datatype ="datetime">40011.56</Proc> |

【客户端调用】

|  |
| --- |
| with TXMLOperator.Create('<Proc/>') do  try  SetElement('Content',  ['Type', 'ProcName', 'Params'],  ['Procedure', UpperCase(ProcName), sParams.DelimitedText],  '');  if BussinessMethodEx(XMLElement.xml, '', sData, sError) < 0 then  begin  hsMsgBox(sError, '执行过程', MB\_ICONWARNING+MB\_OK);  Exit;  end;  with TXMLOperator.Create(sData) do  try  SType := GetElementAttr('', 'datatype');  Value := GetElementValue('');  if SType='number' then  begin  d := StrToFloatDef(Value, 0);  outValue := d;  end  else  if SType='dateTime' then  begin  outValue := hsStrToDateTime(Value);  end  else  outValue := Value;  finally  Free;  end;  Result := true;  finally  Free;  end; |

###### 【GetField】字段取值

**配置说明**

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Content | 接口实现的每个步骤，在这里称为执行内容。  接口的实现由一个或者多个连续的执行内容构成。 | 1属性DBID：当前内容执行的数据库连接ID，对应于数据库连接配置文件HundSun\_DB.xml中的连接ID；缺省为配置中的第一个连接  2属性**Type**：数据字段取值GetField  3属性Condition：内容执行的条件，为布尔型表达式，表达式中引用的参数用“@参数编码”表示，支持数值运算和字符串逻辑判断。  例如Condition="@temp\_c=1"。  [详情见上文“公共配置”]  4属性LoopPath：内容循环执行的XMLPath条件，其中各参数的ValuePath的方法中也必须增加循环序号[#Index]表示参数提取的位置。  例如LoopPath="Item"。  5节点值text()：执行内容的表达式（字符串或者脚本等），格式为数据库操作的脚本  6属性**Field**：表示要获取字段的列名，缺省取第一个字段。  7属性ConcatType：值为"RowConcat"时表示提取字段的多条数据按顺序前后连接，结果输出到sData，该属性缺省则取第一条记录的Field字段。 |

注① 内部取值参数，在Sql语法中定义为以“:”的参数，在存储过程中为输入或者输入输出型参数；注意这里**不是**参数替换，而是作为Sql语法或者存储过程中的参数取值。

内容Content定义参数参数代码.ValuePath，该参数用于该内容的内部提取参数，优先获取参数；如果未定义参数代码.ValuePath，则首先从内部临时参数取值；未找到临时参数，缺省从XML请求参数中的根节点属性取值（属性代码为参数代码）。

IS20130306之前的版本用OraEleDB的驱动，在处理Insert、Select、Update、Delete等查询、执行类的脚本时，不支持参数取值[由于驱动本身不能刷新参数个数和类型]；但可支持对存储过程、函数的参数取值法。

IS20130306及以后的版本，已经完全支持在Insert、Select、Update、Delete、Procedure等的参数取值法。

**示例1**

【用例说明】

数据字段取值，在参数sData中资金输出查询的字段值（不是数据集结果返回）

【GetField配置】

|  |
| --- |
| <Proc ID="CurSysDate" Desc="获取当前系统时间">  <Content Type="GetField" Field="SJ">select to\_char(sysdate,'yyyy-mm-dd hh24:mi:ss') SJ from dual</Content>  </Proc> |

**示例2**

【用例说明】

数据字段取值，将拆分在多条记录的数据合并成一个输出

【GetField配置】

|  |
| --- |
| <Proc ID="Get\_Large\_Data" Desc="获取大数据存储">  <Param Code="DataID" ValuePath="@DataID"/>  <Content Type="GetField" ConcatType="RowConcat">select c\_data from TLargeData where l\_id=@DataID order by l\_xh</Content>  </Proc> |

###### 【UpdateField】更新字段

**配置说明**

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Content | 接口实现的每个步骤，在这里称为执行内容。  接口的实现由一个或者多个连续的执行内容构成。 | 1属性DBID：当前内容执行的数据库连接ID，对应于数据库连接配置文件HundSun\_DB.xml中的连接ID；缺省为配置中的第一个连接  2属性**Type**：更新数据字段UpdateField  3属性Condition：内容执行的条件，为布尔型表达式，表达式中引用的参数用“@参数编码”表示，支持数值运算和字符串逻辑判断。  例如Condition="@temp\_c=1"。  [详情见上文“公共配置”]  4属性LoopPath：内容循环执行的XMLPath条件，其中各参数的ValuePath的方法中也必须增加循环序号[#Index]表示参数提取的位置。  例如LoopPath="Item"。  5节点值text()：执行内容的表达式（字符串或者脚本等），格式为数据库操作的脚本  6属性**Field**：表示更新的数据字段的名称，缺省为第一个字段  7属性Value：表示更新的数据字段的更新值  8属性ValuePath：属性Value不存在时有效，表示更新数据字段的取值方法，取值于请求参数  9属性SplitField：大数据的记录拆分标识，数据拆分后该字段存储记录序号（1..n）  10属性KeyField：与SplitField联合使用，除序号和大数据字段外的其他关键字段，格式为"字段=@参数;字段2=@参数2"，如"l\_id=@DataID"  注：该方法对DB2数据库无效 |

注① 内部取值参数，在Sql语法中定义为以“:”的参数，在存储过程中为输入或者输入输出型参数；注意这里**不是**参数替换，而是作为Sql语法或者存储过程中的参数取值。

内容Content定义参数参数代码.ValuePath，该参数用于该内容的内部提取参数，优先获取参数；如果未定义参数代码.ValuePath，则首先从内部临时参数取值；未找到临时参数，缺省从XML请求参数中的根节点属性取值（属性代码为参数代码）。

IS20130306之前的版本用OraEleDB的驱动，在处理Insert、Select、Update、Delete等查询、执行类的脚本时，不支持参数取值[由于驱动本身不能刷新参数个数和类型]；但可支持对存储过程、函数的参数取值法。

IS20130306及以后的版本，已经完全支持在Insert、Select、Update、Delete、Procedure等的参数取值法。

**示例1**

【用例说明】

更新备注信息，由于备注信息包含特殊字符，不能直接用sql语句递交，需要使用UpdateField方法、或者用Execute方法（带“:”参数的update语句）。

【UpdateField配置】

|  |
| --- |
| <Proc ID="Deposit\_update\_bz" Desc="更新存款\_备注">  <Param Code="AccountID" ValuePath="@AccountID"/>  <Content Type="UpdateField" Field="vc\_bz" ValuePath="text()">  select \* from TZJBD z where l\_id=@AccountID  </Content>  </Proc> |

**示例2**

【用例说明】

大字段数据拆分，根据字段定义的长度限制，将大数据字段拆分为多条记录存储

【UpdateField配置】

|  |
| --- |
| <Proc ID="Update\_Large\_Data" Desc="更新大数据存储" Return="DataID">  <Param Code="DataID" ValuePath="@DataID"/>  <Content Type="Query" Condition="@DataID='' or @DataID=0">  select seq\_tlargedata\_id.nextval NEWID from dual  <Param Code="DataID" ValuePath="METADATA/ROWDATA/ROW/@NEWID"/>  </Content>  <Content Type="UpdateField" SplitField="l\_xh" KeyField="l\_id=@DataID" ValuePath="text()">  select c\_data,l\_xh,l\_id from TLargeData where l\_id=@DataID order by l\_xh  </Content>  </Proc> |

###### 【ImpTable】数据表导入

**配置说明**

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Content | 接口实现的每个步骤，在这里称为执行内容。  接口的实现由一个或者多个连续的执行内容构成。  **注意：**该功能仅在V3.0的三层模式下有效。 | 1属性DBID：当前内容执行的数据库连接ID，对应于数据库连接配置文件HundSun\_DB.xml中的连接ID；缺省为配置中的第一个连接  2属性**Type**：数据导入ImpTable  3属性Condition：内容执行的条件，为布尔型表达式，表达式中引用的参数用“@参数编码”表示，支持数值运算和字符串逻辑判断。  例如Condition="@temp\_c=1"。  4属性 TableName：导入目标表名称 |

**示例**

【用例说明】

将前台的数据集导入到数据库，如本地xls，sqlserver数据库的表、视图数据

【Execute配置】

相对来讲，ImpTable的配置非常简单，只需要配置Type、TableName两个内容即可，condition可选。

|  |
| --- |
| <Proc>  <Content Type="ImpTable" TableName="show2003"/>  </Proc> |

【参数sParam格式】

将前台的数据集转换为Xml格式（通过函数RecordSetToXML进行转换）

###### 【Log】日志记录

**配置说明**

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Content | 插入日志  Content Type="Log" | 1属性**LogType**：日志处理类别。目前支持Field（记录字段值修改差异）  2属性**Value**：日志处理内容（即要记入Log的记录，Sql语句组合生成）  对LogType="Field"，格式为 “操作表名|字段1=值1;字段2=值2;……”，例如： Value="tdictionary|l\_dictionary\_no=@CatID;c\_lemma\_item='@ClassID'"【条件之间用＂；＂隔开】 |

**示例**

【使用场景】

Log一般在一个<Content Type="Execute">的前后记录数据的变化，如果为修改，要记录下这条记录的主键

适应场景：新增、删除、修改

【用例说明】

给定字典类型和字典条目编号，删除该条目。要求记录删除日志。

【Log配置】

|  |
| --- |
| <Proc ID="DicClass\_Del" Desc="删除字典条目">  <Param Code="CatID" ValuePath="@CatID"/>  <Param Code="ClassID" ValuePath="@ClassID"/>  <Content Type="Log" LogType="Field" Value="tdictionary|l\_dictionary\_no=@CatID;c\_lemma\_item='@ClassID'"/>  <Content Type="Execute">delete from tdictionaryL\_DICTIONARY\_NO=@CatID and C\_LEMMA\_ITEM='@ClassID'</Content>  </Proc> |

【调用参数格式】

|  |
| --- |
| <Param CatID="10201" ClassID="8"/> |

###### 【Error】错误提示

**配置说明**

错误提示内容可以包含参数，一般用于校验错误的提示。

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Content | Content Type="Error" | 1属性Condition：内容执行的条件，为布尔型表达式，表达式中引用的参数用“@参数编码”表示，支持数值运算和字符串逻辑判断。  2属性Return：缺省-1 |

**示例**

【说明】

Error主要集成在其他模块基础上，通过< Content Condition >条件判断，如果满足错误条件，通过Return返回-1及错误中文提示信息。

【Error配置】

|  |
| --- |
| <Proc ID="DicClass\_Add" Desc="新增字典条目">  <Param Code="CatID" ValuePath="@CatID"/>  <Param Code="ClassID" ValuePath="@ClassID"/>  <Content Type="Query">select count(\*) C from tdictionary where L\_DICTIONARY\_NO=@CatID and C\_LEMMA\_ITEM='@ClassID'  <Param Code="findcount" ValuePath="METADATA/ROWDATA/ROW/@C"/>  </Content>  <Content Condition="@findcount &gt; 0" Type="Error" Return="-1">字典条目编号重复，已经存在编号为“@ClassID”的条目。请重新设置。</Content>  <Content Type="Log" LogType="Field" Value="tdictionary|l\_dictionary\_no=@CatID;c\_lemma\_item='@ClassID'"/>  <Content Type="Execute">  Insert into tdictionary (L\_DICTIONARY\_NO,c\_lemma\_item,c\_allow\_edit) values (@CatID,'@ClassID','1')  </Content>  <Content Type="UpdateField" Field="vc\_item\_name" ValuePath="@ClassName">  Select \* from tdictionary where L\_DICTIONARY\_NO=@CatID and C\_LEMMA\_ITEM='@ClassID'  </Content>  </Proc> |

###### 【Method】调用方法

**配置说明**

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Param | 自动化接口的参数定义。接口自动化实现时将根据该定义从XML请求参数中提取执行参数。 | 1位置：Proc的子节点，可以有多个参数节点。  2属性Code：参数编码，在该接口中必须是唯一编码。  3属性Value：参数值  4属性ValuePath：属性Value不存在时有效，从XML请求参数中提取值的XPath方法。（注①）  5属性Type：参数值类型，缺省为空，目前用于定义大字段类型，在执行内容为参数法时有效（注②） |
| Content | 方法接口调用  Content Type="Method" | 1属性ProcID：调用的方法的接口编码（即指向XML Proc ID）  2节点值text()：被调用方法的请求参数定义xml，如果为空则参数为当前方法的请求参数。  3属性LoopPath：内容循环执行的XMLPath条件，其中各参数的ValuePath的方法中也必须增加循环序号[#Index]表示参数提取的位置。  例如LoopPath="Item"。 |

注① 参数定义Proc/Param，用于Sql脚本中的带“@”前缀符的参数值替换；注意这里是对Sql语句的字符串替换，需要关注替换后的语法。

注② 参数值类型，目前用于定义大字段类型，在执行内容为参数法时有效。

缺省为空，即对普通的参数替换或者参数值

Blob:Text 将文本转存到Blob参数或者字段

Blob:Hex 将已经转换为16进制文本还原为二进制数据转存到Blob参数或者字段

Blob:File 将文件转存到Blob参数或者字段

**示例**

【用例说明】

费率设置中，如果存在费率则更新（调用更新子过程），如果不存在则新建（调用新建子过程）。

【Method配置】

|  |
| --- |
| <Proc ID="Exchange\_Charges\_Items\_Update" Desc="新增/更新交易所费用定义">  <Param Code="FundID" ValuePath="@FundID"/>  <Param Code="ItemID" ValuePath="@ItemID"/>  <Content Type="Trans" Value="Begin"/>  <Content Type="Method" Condition="@ItemID &gt; 0" ProcID="Exchange\_Charges\_Update"/>  <Content Type="Method" Condition="@ItemID = 0" ProcID="Exchange\_Charges\_Insert"/>  <Content Type="Trans" Finally="" Condition="@Return &lt; 0" Value="Rollback"/>  <Content Type="Trans" Finally="" Condition="@Return &gt;= 0" Value="Commit"/>  </Proc> |

###### 【Trans】执行事务

**配置说明**

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Content | 事务处理  Content Type="Trans" | 1属性**Value**：事务处理类型，可选项有：开始事务BeginTrans、递交事务CommitTrans、取消事务RollbackTrans，或者为简写Begin、Commit、Rollback。 |

调用事务时需要3个内容事务匹配，分别是：开启事务，失败时事务回滚，成功时事务递交。

**示例**

【用例】

删除交易所指定费用类型的费用定义。

【配置】

|  |
| --- |
| <Proc ID="Exchange\_Charges\_Delete" Desc="删除交易所费用定义">  <Param Code="ItemID" ValuePath="@ItemID"/>  <Param Code="FundID" ValuePath="@FundID"/>  <Content Type="Trans" Value="Begin"/>  <Content Type="Log" LogType="Field" Value="tflmx|l\_ztbh=@FundID;l\_zbbh=@ItemID"/>  <Content Type="Execute">  Delete from tflmx m  Where m.l\_ztbh=@FundID and m.l\_zbbh=@ItemID  </Content>  <Content Type="Log" LogType="Field" Value="tfl|l\_bh=@ItemID;l\_ztbh=@FundID"/>  <Content Type="Execute">  Delete from tfl f  f.l\_bh=@ItemID and f.l\_ztbh=@FundID  </Content>  <Content Type="Trans" Finally="" Condition="@Return &lt; 0" Value="Rollback"/>  <Content Type="Trans" Finally="" Condition="@Return &gt;= 0" Value="Commit"/>  </Proc> |

【调用参数格式】

|  |
| --- |
| <Param FundID="5" ItemID="费用编号"/> |

###### 【Tree】树型转换

**配置说明**

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Content | 接口实现的每个步骤，在这里称为执行内容。  接口的实现由一个或者多个连续的执行内容构成。 | 1 Type="Tree" 模块树  2属性TreeField：树结构的生成规则。格式“父节点字段=根节点的父节点值,关键字段”，根节点的父节点值可以为空。  例如TreeField= "VC\_PARENT=0,VC\_CODE"  或者TreeField= "VC\_PARENT=,VC\_CODE" |

**示例**

|  |
| --- |
| <Proc ID="GetSysMenu" Desc="获取当前用户的允许访问的模块树">  <Param Code="AccountState"/>  <Content Type="Query">select sf\_pub\_getgycs(0,'XTCS\_PTSZ\_YYCP') S1, sf\_pub\_getgycs(0,'ZZMKLB') S2, sf\_pub\_getgycs(0,'KHDM') S3 from dual  <Param Code="s1" ValuePath="METADATA/ROWDATA/ROW/@S1"/>  <Param Code="s2" ValuePath="METADATA/ROWDATA/ROW/@S2"/>  <Param Code="s3" ValuePath="METADATA/ROWDATA/ROW/@S3"/>  </Content>  <Content Type="Query"><![CDATA[  Select VC\_SYS, a.VC\_CODE, a.VC\_NAME, a.VC\_PARENT, a.L\_MenuVisible,  \_URL, VC\_PARAM, VC\_DESC  From …  by sysno, a.vc\_parent, a.l\_no  ]]></Content>  <Content Type="Tree" TreeField="VC\_PARENT=0,VC\_CODE"/>  </Proc> |

#### 其他数据服务

通用业务服务是直接配置XML的，而其他数据服务是需要中间件上编码开发来实现的，如SQLLDR服务。

**其他数据服务**仅在V3.0版本的三层模式下支持

##### 调用接口

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **BussinessStream**

create : 2014-06-12, by Paper

description : 其他数据服务接口函数，仅在V3.0版本的三层模式下支持

parameter : streams 业务数据

parameter : sParam 业务传入参数,一般为xml格式

parameter : sData 业务返回结果参数,一般为xml格式

parameter : sError 业务返回错误提示信息

return : 返回值>=0表示成功, 返回值>0的更多业务含义需参照对应业务接口定义

返回值<0表示错误代码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function BussinessStream(streams: array of TStream; const sParam: String;

out sData: OleVariant; out sError: WideString): Integer;

(\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function : **BussinessGetStream**

create : 2014-06-12, by Paper

description : 其他数据服务**下载数据**接口函数，仅在V3.0版本的三层模式下支持

parameter : stream 业务数据

parameter : sParam 业务传入参数,一般为xml格式

parameter : sError 业务返回错误提示信息

return : 返回值>=0表示成功, 返回值>0的更多业务含义需参照对应业务接口定义

返回值<0表示错误代码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

function BussinessGetStream(stream: TStream; const sParam: String;

out sError: WideString): Integer;

###### Oracle数据库SQLLDR服务

**接口说明**

【streams】

给定的文件流，如控制文件、数据文件

【sParam格式】

|  |
| --- |
| <Sqlldr>  <File>sqlldr控制文件.ctl</File>  <Content>control='%s'</Content>  </Sqlldr> |
| <Sqlldr>  <File>sqlldr控制文件.ctl</File>  <File>数据文件.txt</File>  <Content>control='%s' data='%s'</Content>  </Sqlldr> |

【sData返回xml格式的结果】

|  |
| --- |
| <Data>  <File>\tmp\x\x.log</File>  </Data> |
| <Data>  <File>\tmp\x\x.log</File>  <File>\tmp\x\x.bad</File>  </Data> |

sData结果中的文件是指已经产生的文件，如果没有产生该文件，则不生成File节点。文件名称是包含路径的，Windows平台下路径带“\”符号，Linux下带“/”符号。

**示例**

【调用示例1：执行只有控制文件的SQLLDR】

|  |
| --- |
| //上传SHOW2003.ctl文件，并执行SQLLDR  stream1 := TFileStream.Create('F:\SHOW2003.ctl', fmOpenRead);  try  SetLength(streams, 1);  streams[0] := stream1;  iRet := BussinessStream(streams,  '<Sqlldr><File>SHOW2003.ctl</File>'+  '<Content>control=''%s''</Content>'+  '</Sqlldr>',  sData, sError);  if iRet< 0 then  begin  hsWinUtils.hsMsgBox(sError);  Exit;  end  else  Button1.Caption := inttostr(iRet);  finally  stream1.Free;  end;  mData.Text := VarToStr( sData );  //下载执行结果文件，可以区分文件扩展名确认是否下载（这里是全部下载）  with TXMLOperator.Create(sData) do  try  i := 0;  sFile := GetElementValue('File', i);  while sFile<> '' do  begin  stream3 := TFileStream.Create(hsWinUtils.LocalPath+'tmp\'+ExtractFileName(sFile), fmCreate);  try  iRet := BussinessGetStream(stream3, '<GFile>'+sFile+'</GFile>', sError);  if iRet< 0 then  begin  hsWinUtils.hsMsgBox(sError);  Exit;  end;  finally  stream3.Free;  end;  Inc(i);  sFile := GetElementValue('File', i);  end;  finally  Free;  end; |

【调用示例2：执行有控制文件和数据文件的SQLLDR】

|  |
| --- |
| //上传TJK\_JYQS\_REFF03.ctl和reff030522.txt文件，并执行SQLLDR  stream1 := TFileStream.Create('F:\TJK\_JYQS\_REFF03.ctl', fmOpenRead);  stream2 := TFileStream.Create('F:\reff030522.txt', fmOpenRead);  try  SetLength(streams, 2);  streams[0] := stream1;  streams[1] := stream2;  iRet := BussinessStream(streams,  '<Sqlldr><File>TJK\_JYQS\_REFF03.ctl</File>'+  '<File>reff030522.txt</File>'+  '<Content>control=''%s'' data=''%s''</Content>'+  '</Sqlldr>',  sData, sError);  if iRet< 0 then  begin  hsWinUtils.hsMsgBox(sError);  Exit;  end  else  Button1.Caption := inttostr(iRet);  finally  stream1.Free;  stream2.Free;  end;  mData.Text := VarToStr( sData );  //下载执行结果文件，可以区分文件扩展名确认是否下载  with TXMLOperator.Create(sData) do  try  i := 0;  sFile := GetElementValue('File', i);  while sFile<> '' do  begin  stream3 := TFileStream.Create(hsWinUtils.LocalPath+'tmp\'+ExtractFileName(sFile), fmCreate);  try  iRet := BussinessGetStream(stream3, '<GFile>'+sFile+'</GFile>', sError);  if iRet< 0 then  begin  hsWinUtils.hsMsgBox(sError);  Exit;  end;  finally  stream3.Free;  end;  Inc(i);  sFile := GetElementValue('File', i);  end;  finally  Free;  end; |

##### 其他数据服务开发说明（C++）

其他数据服务客户端无需开发，直接调用标准接口即可。

中间件需要开发服务实现类，继承TFS\_FileProc数据流服务抽象类，并做服务注册。数据流服务需要实现下面的接口：

virtual int Execute(TDS\_Session \* lpSession, IRCfgObj \* lpFiles, TDS\_XmlParam \* lpXmlParam)=0;

参数：lpSession短会话，lpFiles数据流XML，lpXmlParam服务参数

返回值：>=0成功，<0失败

【lpFiles数据流XML格式】

|  |
| --- |
| <Files Count="2" Dir="x">  <File Index="0">\tmp\x\filename1.a</File>  <File Index="1">\tmp\x\filename2.b</File>  </Files> |

【lpXmlParam服务参数】

服务参数由该服务来确定，例如上述的SQLLDR服务的sParam参数格式

### 清算模块－数据导入配置

#### TStoreType表结构说明

|  |  |  |
| --- | --- | --- |
| TStoretype | | |
| **字段名称** | **属性** | **说明** |
| L\_ID | NUMBER(4) not null | 序号 |
| C\_SORUCE\_TYPE | VARCHAR2(10) not null | 数据源类别，对应数据字典表：60001  ztz：普通账套从账套组接收数据 |
| VC\_TBLNAME | VARCHAR2(30) | 目的表名称（恒生系统库表） |
| C\_FILE\_FORMATTYPE | VARCHAR2(1) not null | 处理类别：  1:ADO文件导入(\*.dbf/\*.xls); 2:SqlLoad方式导入(\*.txt); 3:DB导入(O3或者三方数据库)[目前支持SQL,ORACEL];  4:XML文件导入 |
| VC\_DATA\_CTRL | VARCHAR2(4000) | 数据导入控制文件【对支持ADO接口的，为驱动配置和提取脚本的配置XML；对sqlldr方式的，为提取脚本或者包含提取脚本的xml；  对DB导入，为配置的连接编号和取数sql；对xml文件导入，为调用数据服务引擎的配置xml 】 |
| C\_DATA\_TYPE | CHAR(1) default '1' | 数据类别：  1：基本面数据；  2：行情数据；  3：交易数据；  4：TA数据  5：对照数据  7：权益公告 |
| VC\_CAPTION | VARCHAR2(50) | 接口名称[界面清算步骤显示名称] |
| VC\_FILENAME | VARCHAR2(50) | 导入的文件名称 |
| VC\_TYPE | VARCHAR2(10) |  |
| VC\_VERSION | VARCHAR2(50) |  |
| VC\_BZ | VARCHAR2(200) | 备注 |
| VC\_ZZMK | VARCHAR2(200) |  |

**【配置说明：文件路径和文件名称的通配符】**

在上述C\_FILE\_FORMATTYPE=1,2,4中的文件导入类型中，在配置文件路径和文件名称中允许基金代码匹配[$FUND\_CODE] 、[日期格式]匹配和“\*”、“?”等文件通配符的匹配，在C）叙述的控制格式的Sql脚本中，也支持[日期格式]参数。

[日期格式]支持包含“Y”“M”“D”“A”四个大小写字符和加减数字，如：

|  |  |
| --- | --- |
| 示例 | 说明 |
| [YYYYMMDD] 或者 [yyyymmdd] | 在2008-7-9时，匹配为“20080709”  在2008-10-31时，匹配为“20081031” |
| [YYYYMD] 或者 [yyyymd] | 在2008-7-9时，匹配为“200879”  在2008-10-31时，匹配为“20081031” |
| [yyyymmdd-1] | 在2008-7-9时，匹配为“20080708”  在2008-10-31时，匹配为“20081030” |
| [yyyymmdd+1] | 在2008-7-9时，匹配为“20080710”  在2008-10-31时，匹配为“20081101” |
| [YYYYaDD]或者[yyyyadd] | 在2008-7-9时，匹配为“2008709”  在2008-10-31时，匹配为“2008a31”  在2008-12-31时，匹配为“2008c31” |
| [YYYYADD]或者[yyyyAdd] | 在2008-7-9时，匹配为“2008709”  在2008-10-31时，匹配为“2008A31”  在2008-12-31时，匹配为“2008C31” |
| [yyyyadd-1] | 在2008-7-9时，匹配为“2008708”  在2008-10-31时，匹配为“2008a30” |
| [yyyy-mm-dd] | 在2008-7-9时，匹配为“2008-07-09”  在2008-10-31时，匹配为“2008-10-31” |
| [yyyy]年[m]月[d]日 | 在2008-7-9时，匹配为“2008年7月9日”  在2008-10-31时，匹配为“2008年10月31日” |
| [yyyy年m月d日] | 错误格式 |

**【SQL脚本和SQLLDR脚本中的参数】**

SQL脚本和SQLLDR脚本中支持 [$FUND\_ID]、[$FUND\_CODE]、[日期格式]三个参数，分别表示文件路径、文件名称、基金代码和日期，其中日期是包含ymd的格式化字符串，如[YYYYMMDD]。

#### 配置说明

##### OLEDB接口方式导入ADO文件数据库

C\_FILE\_FORMATTYPE=1 (ExecData\_ADOFile函数) [ADO文件导入(\*.dbf/\*.xls)]

该接口类别支持缺省配置，即当导入文件为标准格式时，VC\_DATA\_CTRL可以为空，完整的 VC\_DATA\_CTRL配置包括驱动（ADO连接字符串）和内容（sql脚本）两个部分。

###### 导入Excel文件

  当文件后缀名为“.xls”时，其格式如下：

|  |
| --- |
| <data>  <Provider>  Provider=Microsoft.Jet.OLEDB.4.0;Data Source="[$FILE\_PATH][$FILE\_NAME]";Extended Properties="Excel 8.0;HDR=NO;IMEX=1;"  </Provider>  <Content><![CDATA[  select F1 as effective\_date, F2 as securities\_code,  F3 as securities\_name, F4 as securities\_name\_en,  F5 as exchange, F6 as industry\_code,  F7 as industry\_name, F8 as industry\_name\_en,  F9 as industry\_code\_full, F10 as industry\_name\_full,  F11 as industry\_name\_full\_en  from [$A2:IV65535]  where trim(F1)<>''  ]]></Content>  </data> |

如果导入的EXCEL是标准格式，那么可以使用如下缺省规则

|  |  |
| --- | --- |
| **实际配置项中没有提供如下配置项** | **缺省配置项** |
| <Provider> Provider=Microsoft.Jet.OLEDB.4.0;Data Source="[$FILE\_PATH][$FILE\_NAME]";Extended Properties="Excel 8.0;HDR=NO;IMEX=1;"  </Provider> | <Provider>  Provider=Microsoft.Jet.OLEDB.4.0;Data Source="[$FILE\_PATH][$FILE\_NAME]";Extended Properties="Excel 8.0;HDR=YES;IMEX=1;"  </Provider> |
| <Content>  …..  </Content> | 第一行列名，第二行开始数据；没有需要排除的数据行  <Content>  select \* from [$A1:IV65535]  </Content> |

|  |  |
| --- | --- |
| **Excel格式配置项中各元素的含义** | |
| **元素** | **描述** |
| data | 数据开始格式 |
| <Provider>  </Provider> | 提供数据连接的连接字符串。  如果使用标准格式，该值可缺省 |
| [$FILE\_PATH] | 载入的文件的路径 |
| [$FILE\_NAME] | 载入文件的名称 |
| Extended Properties | 打开Excel的版本连接模式 |
| HDR | YES：第一行是标题，不做为数据使用  NO： 第一行不是标题，做为数据来使用  表示在select的结果中，是从第一行还是第二行数据开始取值 |
| IMEX | 0：汇出模式，只能写入  1：汇入模式，只能读取  2：链接模式，可读取和写入  [在本系统中都设置1] |
| <Content>…..  </Content> | 执行内容，SQL脚本  允许缺省，如果缺省则使用缺省配置项 |
| from [$A1:IV65535] | 表示从载入的Excel文件中从第二行开始读取数据，第一行为字段名称 |
| from [$A2:IV65535] | 表示从载入的Excel文件中从第三行开始读取数据，第二行为字段名称 |
| from [Sheet1$A2:IV65535] | 表示读取Excel中[Sheet1]这个Sheet页，同时从第三行开始读取数据，第二行为字段名称  如果导入的Excel只有一个Sheet页 ，那么Sheet1可省略 |
| F1 as effective\_date | 该模式在没有Excel的列名情况下才使用。F1对应接口表的effective\_date字段  如果Excel有列名并且和估值系统接口表的列同名，那么可以不用as，否则用as |
| where trim(F1)<>'' | 如果载入的Excel中F1列有空值（不合法值），那么需要加where条件排除，否则不用加 |

###### 导入DBF文件

**注意：**一般不使用OLEDB方式导入DBF文件，而是用SQLLDR导入

除文件后缀名为“.xls”外，其它文件类型都被当作DBF文件来处理。如果控制格式为空，将缺省匹配为DBF导入格式，默认格式为（目前系统中都是用默认值）：

|  |
| --- |
| <data>  <Provider>Provider=VFPOLEDB.1;Data Source="[$FILE\_PATH]"; Mode=Share Deny None;Extended Properties="";User ID=;Password=; Mask Password=False;Cache Authentication=False; Encrypt Password=False;Collating Sequence=MACHINE;DSN=""</Provider>  <Content>select \* from [$FILE\_NAME]</Content>  </data> |

|  |  |
| --- | --- |
| **实际配置项中没有提供如下配置项** | **缺省配置项** |
| <Provider>  Provider=VFPOLEDB.1;Data Source="[$FILE\_PATH]"; Mode=Share Deny None;Extended Properties="";User ID=;Password=; Mask Password=False;Cache Authentication=False; Encrypt Password=False;Collating Sequence=MACHINE;DSN=""  </Provider> | <Provider>  Provider=VFPOLEDB.1;Data Source="[$FILE\_PATH]"; Mode=Share Deny None;Extended Properties="";User ID=;Password=; Mask Password=False;Cache Authentication=False; Encrypt Password=False;Collating Sequence=MACHINE;DSN=""  </Provider> |
| <Content>  …..  </Content> | <Content>  select \* from [$FILE\_NAME]  </Content> |

当DBF格式不符合缺省格式要求时，如有字段名称不匹配、数据行需要过滤等情况，需要提供配置项，例如：（深交所行情—sjshq，在Sql脚本中去除000000非行情数据）

|  |
| --- |
| <data>  <Content>select \* from [$FILE\_NAME] where hqzqdm<>'000000'</Content>  </data> |

|  |  |
| --- | --- |
| **DBF格式配置项中各元素的含义** | |
| **元素** | **描述** |
| data | 数据开始格式 |
| <Provider>  </Provider> | 提供数据连接的连接字符串。  允许缺省，如果缺省则使用缺省配置项 |
| Provider=VFPOLEDB.1 | 表示数据库支持Visual FoxPro |
| Data Source=”[$FILE\_PATH]” | 载入文件的数据源和文件所在路径 |
| [$FILE\_NAME] | 载入文件的名称 |
| Mode | Connection 对象修改数据的权限  READWRITE：表明权限为读/写。 Share Deny None：允许其他人以任何权限打开连接，不拒绝其他人的读或写访问 |
| Extended Properties | 打开Excel的版本连接模式，DBF这里不用设置 |
| <Content>…..  </Content> | 执行内容，SQL脚本  允许缺省，如果缺省则使用缺省配置项 |
| User ID | 访问dbf的用户名 |
| Password | 访问dbf的密码 |
| Mask Password | 密码是否可见 |
| Cache Authentication | 是否缓存验证 |
| Encrypt Password | 是否对密码加密 |
| Collating Sequence |  |
| DSN | 连接的dsn源 |

##### SQLLoad导入格式化文本文件

C\_FILE\_FORMATTYPE=2

该接口类别**不支持**缺省配置，必须按照《oracle sqlldr控制文件模板》的要求，根据导入文件格式的定义进行配置。

  示例1：（导入接口配置：上交所权证信息qzxx[MMDD].txt）

|  |
| --- |
| LOAD DATA  Append INTO TABLE QZXX(  XQDM POSITION(1:6) CHAR,  QZDM POSITION(8:13) CHAR,  BDZQ POSITION(15:20) CHAR,  QZLX POSITION(22:25) CHAR,  QZYE POSITION(27:43) DECIMAL EXTERNAL,  JSFS POSITION(45:48) CHAR,  XQFS POSITION(50:53) CHAR,  JSJ POSITION(55:74) DECIMAL EXTERNAL,  XQJG POSITION(76:83) DECIMAL EXTERNAL,  XQBL POSITION(85:90) DECIMAL EXTERNAL,  DQRQ POSITION(92:99) CHAR,  SYRQ POSITION(101:108) CHAR  ) |

|  |  |
| --- | --- |
| **元素** | **描述** |
| data | 数据开始格式 |
| Append..QZXX | 1：Append：向表Qzxx追加记录  2：INSERT：装载空表 如果原先的表有数据 sqlloader会停止默认值  3：REPLACE：原先的表有数据 会全部删除  4：TRUNCATE：原先表有数据，那么全部删除同时释放空间 |
| POSITION(1:6)  fields terminated by '|'  fields terminated by ',' | 取第一位到第六位的内容赋给XQDM； |
| 如果没有用位置装载数据，那么数据之间可能是用‘|’或’,’来分割的 |
| [$FUND\_ID] | 账套编号 |
| [YYYYMMDD] | 系统传入的清算日期 |

   示例2：（导入接口配置：上交所行情信息show2003.dbf）

|  |
| --- |
| LOAD DATA  INFILE \*  Append INTO TABLE SHOW2003(  DELFLAG POSITION(1:1) CHAR,  S1 POSITION(2:7) CHAR,  S2 POSITION(8:15) CHAR,  S3 POSITION(16:23) DECIMAL EXTERNAL,  S4 POSITION(24:31) DECIMAL EXTERNAL,  S5 POSITION(32:43) DECIMAL EXTERNAL,  S6 POSITION(44:51) DECIMAL EXTERNAL,  S7 POSITION(52:59) "sf\_pub\_strtonum(:s7)",  S8 POSITION(60:67) "sf\_pub\_strtonum(:s8)",  S9 POSITION(68:75) "sf\_pub\_strtonum(:s9)",  S10 POSITION(76:83) "sf\_pub\_strtonum(:s10)",  S11 POSITION(84:93) "sf\_pub\_strtonum(:s11)",  S13 POSITION(94:101) "sf\_pub\_strtonum(:s13)",  S15 POSITION(102:111) "sf\_pub\_strtonum(:s15)",  S16 POSITION(112:119) "sf\_pub\_strtonum(:s16)",  S17 POSITION(120:129) "sf\_pub\_strtonum(:s17)",  S18 POSITION(130:137) "sf\_pub\_strtonum(:s18)",  S19 POSITION(138:147) "sf\_pub\_strtonum(:s19)",  S21 POSITION(148:157) "sf\_pub\_strtonum(:s21)",  S22 POSITION(158:165) "sf\_pub\_strtonum(:s22)",  S23 POSITION(166:175) "sf\_pub\_strtonum(:s23)",  S24 POSITION(176:183) "sf\_pub\_strtonum(:s24)",  S25 POSITION(184:193) "sf\_pub\_strtonum(:s25)",  S26 POSITION(194:201) "sf\_pub\_strtonum(:s26)",  S27 POSITION(202:211) "sf\_pub\_strtonum(:s27)",  S28 POSITION(212:219) "sf\_pub\_strtonum(:s28)",  S29 POSITION(220:229) "sf\_pub\_strtonum(:s29)",  S30 POSITION(230:237) "sf\_pub\_strtonum(:s30)",  S31 POSITION(238:247) "sf\_pub\_strtonum(:s31)",  S32 POSITION(248:255) "sf\_pub\_strtonum(:s32)",  S33 POSITION(256:265) "sf\_pub\_strtonum(:s33)"  )  BEGINDATA  [$DbfToTxt] |

本例中“[$DbfToTxt]”为数据替换符，在运行时将被替换为由DBF数据转换成的文本（文本行数等于DBF记录数，每行以换行符结束）。

|  |  |
| --- | --- |
| **元素** | **描述** |
| data | 数据开始格式 |
| Append..SHOW2003 | 1：Append：向表Show2003追加记录  2：INSERT：装载空表 如果原先的表有数据 sqlloader会停止。默认值  3：REPLACE：原先的表有数据 会全部删除，替换成新装载的记录 4：TRUNCATE：原先表有数据，那么全部删除同时释放空间 |
| S1 [$POS:S1] | 首先需要根据一定规则把” [$POS:”替换成” POSITION(\*:20)”的格式，然后再在对应位置取值赋给S1字段 |
| Error ErrCode | 用来判断当前表中是否有当前日期以外的数据，存在的话，就提示错误 |
| INFILE \* | 要导入的内容，BEGINDATA后面就是导入的内容  INFILE 't.dat' ：要导入的文件 |
| BEGINDATA | 对应开始的 INFILE \* 要导入的内容 |
| [$DbfToTxt] | 用来标记前面有多少行数据要导入标记，同时也是读入数据的位置 |
| sf\_pub\_strtonum | 字符串转换成数字，如果串为空，那么返回null |
| LocalParam Code="ID" | 当前操作先取值，然后赋给下面的变量@ID |

   示例3：（导入接口配置：中金所期货交易数据）

|  |
| --- |
| <data><Content><![CDATA[  OPTIONS (skip=4)  LOAD DATA  Append INTO TABLE TJK\_ZJSQS\_TRADE(  JYHYH POSITION(1:10) CHAR,  KHBM POSITION(11:22) CHAR,  HY POSITION(23:32) CHAR,  CHBH POSITION(33:44) CHAR,  CJL POSITION(45:52) DECIMAL EXTERNAL,  CJJ POSITION(53:64) DECIMAL EXTERNAL,  CJE POSITION(65:78) DECIMAL EXTERNAL,  CJSJ POSITION(81:90) CHAR,  MM POSITION(91:96) CHAR,  KP POSITION(97:103) CHAR,  BDH POSITION(104:117) CHAR,  XWH POSITION(118:131) CHAR,  L\_ZTBH CONSTANT "[$FUND\_ID]",  D\_YWRQ EXPRESSION "TO\_DATE([YYYYMMDD],'YYYYMMDD')"  )]]></Content>  <Execute><![CDATA[  update TJK\_ZJSQS\_TRADE  set hy = (select vc\_isin  from tzqxx  where (l\_zqlb = 6 or (l\_zqlb =7 and l\_zqlbmx1=1)) and  upper(vc\_isin) = upper(hy) and  vc\_isin != hy  )  where d\_ywrq = to\_date([YYYYMMDD],'yyyymmdd') and  l\_ztbh = [$FUND\_ID] and  exists(select 1  from tzqxx  where (l\_zqlb = 6 or (l\_zqlb =7 and l\_zqlbmx1=1)) and  upper(vc\_isin) = upper(hy) and  vc\_isin != hy  )  ]]></Execute>  </data> |

|  |  |
| --- | --- |
| **元素** | **描述** |
| OPTIONS | 标识符号 |
| (skip=4) | 取数据时跳过前面4行，从第5行开始取数 |
| FIELDS TERMINATED BY WHITESPACE | 没有用位置装载数据，数据之间用‘空格’来分割 |
| [$FUND\_ID] | 账套编号 |
| [YYYYMMDD] | 系统传入的清算日期 |

##### ADODB方式从外部数据库导入

###### 数据源为Sql Server / DB2等

目前系统中使用<Content></Content>获取sql语句，赋给tpsADOData.commandText，然后使用函数ExecData\_ADO把数据写到本地数据接口表目标对象，也是客户端tpsADOData控件，最后执行UpdateBatch保存数据。其配置格式如下

|  |
| --- |
| <data>  <Provider DBConnection="数据库ID"/>  <Content><![CDATA[  select zcdy, sqrq, qrrq, jyfx, zqdm, zqmc, tzlx, ccfl, dyzqdm, zyzqmc, dysc,  sxf, sl, cjzt, bz, zjhm, zjzh, [$FUND\_ID] l\_ztbh, to\_date([YYYYMMDD], 'YYYYMMDD') d\_ywrq  from bondtransfer  where sqrq = to\_date([YYYYMMDD], 'YYYYMMDD')  ]]></Content>  </data> |

或者如下

|  |
| --- |
| <data>  <LocalParam Code="beginrq">select to\_char(sf\_pub\_addtradedays(nvl(sf\_pub\_getgycs([$FUND\_ID], 'JJJYRLX'),'1'),to\_date([YYYYMMDD],'yyyymmdd'),-1,2)+1,'yyyymmdd') rq from dual </LocalParam>  <LocalParam Code="endrq">select [YYYYMMDD] rq from dual </LocalParam>  <Provider DBConnection="数据库ID"/>  <Content><![CDATA[ select zqdm, net\_value, begin\_net, total\_net, adjust\_net, adjust\_int, div\_date, data\_date  from par\_openfund\_info  where data\_date >= '@beginrq' and  data\_date <= '@endrq']]>  </Content>  </data> |

其中，数据库ID对应于数据库连接配置文件HundSun\_DB.xml中的连接ID；SQL脚本中支持 [$FUND\_ID]、 [YYYYMMDD]参数，分别表示账套ID和日期。

|  |  |
| --- | --- |
| **元素** | **描述** |
| Provider DBConnection | 数据库连接标准格式 |
| 数据库ID | 对应于数据库连接配置文件HundSun\_DB.xml中的连接ID |
| <Content>  </Content> | 查询sql脚本 |
| [$FUND\_ID] | 当前账套编号 |
| [YYYYMMDD] | 清算操作的日期 |
| <Execute>  </Execute> | 执行的sql脚本 |
| LocalParam Code= beginrq | 表示先获取日期，存入变量beginrq，做为参数传给后面的变量@beginrq |
| @beginrq | 该值有前面的LocalParam Code传入 |
| sf\_pub\_addtradedays | 估值系统中存在的函数 |

（针对下面的“数据库ID”,位于系统目录下的HundSun\_DB.xml文件）：

|  |  |
| --- | --- |
| DBConnection ID= | 连接标识符 |
| "1" | 连接本地自己oracle数据库 |
| "2" | 风控数据库(SQL) |
| "3" | 融资融券数据库(SQL) |
| "ZG" | 资产管理数据库(SQL) |
| "ZRZY" | 金蝶财务数据库(SQL) |
| "HTSJZX" | 华泰数据中心(Oracle) |
| "ZHGLPT" | 综合管理平台数据库(Oracle) |

注意：后续配置直连资管接口（DBConnection="ZG"）都需要小写表名和字段名。资管sqlserver不支持大写，大写的情况下估值清算报错，表名无效。

###### 数据源为Oracle

对于数据源是Oracle的连接模式，如果数据源与FA数据库不是同一个数据库实例，需要先在本地oracle数据库中建立DBLink，再建立对应表的同义词。如果是同一个数据库实例，需要对数据源的相关表做授权，然后建立对应表的同义词。

一般直接使用<Execute></Execute>，把数据直接insert到本地对应接口表，其格式事例如下：

|  |
| --- |
| <data>  <Provider DBConnection="1"/>  <Execute><![CDATA[  delete from tjk\_zgqs\_tzbdzc a  where a.l\_ztbh = [$FUND\_ID] and  a.d\_ywrq = to\_date([YYYYMMDD],'yyyymmdd')  ]]></Execute>  <Execute><![CDATA[  insert into tjk\_zgqs\_tzbdzc  (RQ, JJDM, TZDM, ZQDM, JYSCBH, LB, SL, JE, XSJJRQ, L\_ZTBH, D\_YWRQ)  select a.RQ, a.JJDM, a.TZDM, a.ZQDM, a.JYSCBH, a.LB, a.SL, a.JE, a.XSJJRQ, [$FUND\_ID] l\_ztbh,a.d\_ywrq  from (select \* from tjk\_zgqs\_tzbdzc c  where c.l\_ztbh in (select l\_tjzt from ttjzt\_map  where l\_ztbh =  decode((select vc\_ztlx from tsysinfo where l\_id = (select l\_sjzt from tsysinfo where l\_id = [$FUND\_ID])),  3,  (select l\_sjzt from tsysinfo where l\_id = [$FUND\_ID]),  [$FUND\_ID]))) a  where a.d\_ywrq = to\_date([YYYYMMDD],'yyyymmdd')  ]]></Execute>  </data> |

|  |  |
| --- | --- |
| **元素** | **描述** |
| Provider DBConnection | 数据库连接标准格式 |
| "1" | 对应于数据库连接配置文件HundSun\_DB.xml中的连接ID |
| < Execute >  </ Execute > | 数据库执行脚本 |
| [$FUND\_ID] | 当前账套编号 |
| [YYYYMMDD] | 清算操作的日期 |

##### XML文件导入

C\_FILE\_FORMATTYPE=4 (ExecData\_XmlFile函数)

该模式支持账套ID [$FUND\_ID]参数。

|  |
| --- |
| <Proc DataPath="result[@name= &quot; CLIENTMISFEE &quot;]" Desc="客户费用">  <Param Code="AMOUNT" ValuePath="CLIENTMISFEE[#Index]/@AMOUNT"/>  <Param Code="CLIENTID" ValuePath="CLIENTMISFEE[#Index]/@CLIENTID"/>  <Content Type="Execute" LoopPath="CLIENTMISFEE">  insert into TJK\_SGE\_CLIENTMISFEE(L\_ZTBH,D\_YWRQ,AMOUNT,CLIENTID)  values([$FUND\_ID],to\_date([YYYYMMDD],'yyyymmdd'), '@AMOUNT','@CLIENTID')  </Content>  </Proc> |

载入文件后调用BussinessMethodEx接口

|  |  |
| --- | --- |
| **元素** | **描述** |
| DataPath | 读数据的开始位置 |
| result[@name=";CLIENTMISFEE";] | result是xml文件的节点，表示读取xml中节点名称为result且符合条件（属性name="CLIENTMISFEE"）的数据 |
| Param Code | 读取指定路径的数据赋给参数AMOUNT |
| Content Type | Sql操作类型（查询还是执行） |
| LoopPath="CLIENTMISFEE" | 按"CLIENTMISFEE"为节点做循环Content Type操作 |
| [$FUND\_ID] | 当前清算的账套编号 |
| [YYYYMMDD] | 清算的具体日期 |
| &quot; | 双引号符号标识“"” |

### 报表开发

【报表模块的程序结构图】



【报表查询的程序流程】

1. 前台根据报表名称读取配置文件，根据配置生成查询参数和载入报表模板；
2. 数据服务引擎根据配置文件查询并返回一个或者多个数据集；
3. 显示/打印报表

#### 报表功能概述

【Excel文件导出】

以微软Excel97二进制文件格式规范生成导出文件，与Excel完全兼容。

【数据处理方法】

目前报表已经以下几种取值方法：

1. 支持单元格函数取值，返回数值或者文本；
2. 支持单元格数学运算，返回数值；
3. 支持从结果数据集取数（单行或者多行表格）；
4. 支持交叉表运算（行列同时展开的特殊表格）；
5. 支持多数据集功能；
6. 支持行分组功能和列分组功能；
7. 支持单元格显示查询参数。

【打印功能】

支持页头、页尾、页码格式，页面设置（页面大小、边框、方向）。

【菜单入口】

报表菜单模块入口URL格式为“hsReport.dll|ShowReportForm|报表编号”

#### 报表开发工具－模板设计器

##### 参数和函数表达式

【替换形参数】

参数取值采用参数替换符号"@"，即"@参数代码"

报表系统提供缺省的五个替换参数：

报表名称 @Title

公司名称 @CorpName,

用户名称 @UserName

账套名称 @FundName

发格业务日期 @Date

其中，账套和日期需要根据参数是否显示来确定，如果不显示则参数无效

如自定义参数Code="BeginDate"在模板中的配置格式为 @BeginDate

【取值型参数】

取值型参数方法有四个：

= Title()

= CorpName()

= UserName()

= PARAM("FundName")

Param()是用于获取报表定义的查询参数的通用函数，需要传入参数名称。

【常用函数】

提供Excel的常用函数和数学表达式处理。

【取值函数】

根据Texpression表的定义，提供取值函数。

##### 配置参数和取数方法

目前报表的客户端配置支持两种类型：

1. 报表的所有数据项都通过函数表达式的方式获取，不需要提供查询结果数据集。该类型只需要将报表模板放入Rpt目录下，配置报表菜单[模块入口](#_菜单发布)，并保证菜单入口的报表标题与报表模板名称一致。
2. 报表的部分数据或者全部数据由查询结果数据集提供，即报表中包含表格数据。该类型报表需要定义报表配置xml，配置方法在 “[数据服务引擎](#_数据服务引擎（推荐使用）)”基础上增加了报表标题匹配和动态参数两项配置方法。

在报表模板设计界面中，数据→XMLDATA打开配置窗口：

|  |
| --- |
| <Proc ID="Test2" Caption="Report Test2">  <Param Code="FundID" ValuePath="@FundID" BookType="统计账套+基金账套" Single="false"/>  <Param Code="Date" ValuePath="@Date"/>  <Content Type="Query">  a.\*, z.vc\_zqjc, z.l\_zqlb, decode(z.l\_zqlb,1,'股票',2,'债券',3,'回购',4,'基金','其他') vc\_zqlb, s.vc\_name  from taccountzqjc a, tsysinfo s, tzqxx z  where a.d\_date = to\_date('18991230','yyyymmdd')+@Date and  a.l\_ztbh in (@FundID) and  a.l\_ztbh=s.l\_id and  a.l\_zqnm=z.l\_zqnm  order by a.l\_ztbh, z.l\_zqlb, z.vc\_zqdm  </Content>  </Proc> |

报表引擎配置格式说明：

|  |  |  |
| --- | --- | --- |
| **元素** | **描述** | **格式定义** |
| Proc | 报表配置节点 | 1位置：根节点  2属性ID：报表编号。  3属性Caption：报表标题，可选项。报表标题优先使用报表参数“BT\_报表代码”的参数值，第二选择为该属性，第三选择为最小的模板名称。 |
| Param | 自动化接口的参数定义。设置属性**Caption**后支持报表动态参数。 | 1位置：Proc的子节点，可以有多个参数节点。  2属性**Code**：参数编码，在该接口中必须是唯一编码。  3属性**Caption**：动态参数的标题，可显示参数在界面上  4属性ValuePath：属性Value不存在时有效，从XML请求参数中提取值的XPath方法，为动态参数时要求格式为“@参数编码”。  5属性Type：参数类型，为动态参数时需要给定参数类型，缺省为字符串。这里提供ptString, ptInt, ptFloat, ptDate, ptTime, ptBoolean, ptEnum, ptEnumSQL共8种参数类型，分别代表了字符串、整形、浮点数、日期、时间、布尔型、下拉框、根据Sql提供下拉框内容。  6属性Single：单选多选标志，true单选、false多选，缺省为单选  7属性BookType：对账套参数有效，表示选择账套的下拉内容，设置的值选项详见账套控件说明。账套参数要求配置取值方法为：ValuePath="@FundID" |
| Content |  | 1位置  2属性Type：查询的方法类别，支持Query和Procedure |
| 数据查询  Content Type="Query" | 1节点值text()：执行内容的表达式（字符串或者脚本等），格式为数据库操作的脚本  2属性ResultType：输出的结果类型，缺省类型为"XML"，即输出的是XML格式编码的字符串。设置为"RecordSet"时，输出的是ADOInt.RecordSet，可以这样使用：  rs := Recordset(TVarData(sData).VDispatch); |
| 内容查询  Content Type="Procedure" | 1属性**ProcName**：表示存储过程的名称  2属性**Params**：存储过程参数  3属性Return：存储过程返回参数中，表示错误代码的参数名称  4属性Error：存储过程返回参数中，表示错误信息的参数名称 |

##### 设计报表『页面』

【定义数据项】

数据项：在本报表设计器中，数据项定义了具备一定业务逻辑的数据处理方法和显示区域，如表格、交叉表、子报表等。

数据项类别：

* 常规

在该区域内，可以设置任何数据集的字段，但**不会**根据数据集的记录数生成多行数据。

* 数据列表

在该区域内，可以设置任何数据集的字段，并以关联数据集的记录数生成多行数据，即生成数据表格。

可支持行分组和列分组功能

* 数据汇总行 （暂不支持）

根据关联数据集的所有记录生成汇总数据，如合计、平均、取个数等。

* 交叉表

生成横向和纵向都能够按照指定字段的数据维度展开的数据表格，详见[交叉表配置](#_交叉表配置)。

* 内嵌子报表 （暂不支持）

报表中的子报表，用该功能可以实现复杂的报表排版。

数据项有四个操作功能：

数据项列表：查看已经设置的所有数据项设置信息（数据区域、数据项类别、关联数据集）。

设置数据项：查看当前选定单元格所在的数据项。

增加数据项：将当前选定单元格范围设置为一个新建的数据项。

删除数据项：将当前选定单元格对应的数据项删除。

查看和删除数据项时，设计器会将当前数据项区域用选择方框展示出来。

【打印－页面设置】

表头、表尾、页头、页尾。

表头、表尾分别在首尾页打印，页头、页尾在每页打印。

页面设置。

设置打印页面的页面大小、边框、页面方向。

页码格式。

设置是否显示页码，页码的位置、页码的显示格式。

候选的显示格式有“第 @Page 页 共 @PageCount 页”等五项，也可以自定义。其中@Page表示当前打印的页码，@PageCount表示页码总数。

##### 分组报表配置

在普通“数据列表”的基础上，可支持分组报表的功能。分组报表可支持行分组和列分组。

**行分组：**对同一类的数据插入汇总行，配置格式为

[RGroup:(field1,field2,…)fieldx]

小括号里面的为分组的字段或者字段列表，后面为显示的字段

分组行的数据，除了分组单元格外，其他单元格上可填写固定的数据、字段、汇总表达式。对汇总表达式，可实现该分组内的数据全部统计。

分组行可位于明细数据上方或者下方，所有的分组行必须在“数据列表”定义的区域内。

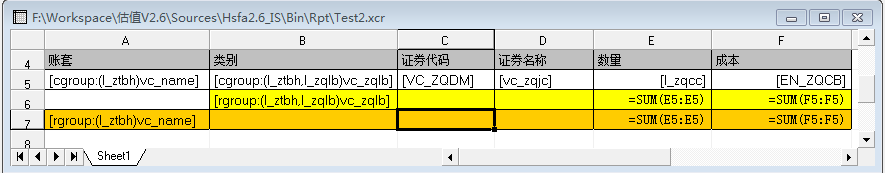
**列分组：**对同一类的数据生成合并单元格，配置格式为

[CGroup:(field1,field2,…)fieldx]

小括号里面的为分组的字段或者字段列表，后面为显示的字段

分组列位于明细数据的行，必须在“数据列表”定义的区域内。

同一个“数据列表”定义的区域内，可同时支持多个分组行和分组列。如：



运行效果：



##### 交叉表配置

【配置方法】

选定需要定义交叉表的单元格区域（示例中为3×3），设置数据项类型为交叉表，并设置边框、字体、显示格式等。

交叉表区域的单元格有4种类别：交叉计算单元格、交叉汇总单元格、字段取值单元格、和静态文本。

整个报表中仅仅支持一个交叉计算单元格，配置格式为“[Cross:xfield,yfield,value]”，其中xfield为交叉算法的横向字段名称，yfield为交叉算法的纵向字段名称，value为交叉算法的数据取值字段。如上面例子，按账套ID和销售商ID来统计申购金额，配置为“[cross:l\_ztbh,vc\_jgdm,en\_sgje]”。

交叉表区域中的交叉汇总单元格和字段取值单元格必须在交叉计算单元格的正上方、正下方、正左侧、正右侧的其中之一。交叉汇总单元格支持3中统计方法：合计、平均、取个数，格式分别是“[sum()]”、“[average()]”或者“[avg()]”、“[count()]”。字段取值单元格可以显示交叉后的任何字段的值，格式为“[fieldname]”，其中fieldname为取值字段名称。

静态文本可以是任何文本（“=”开头、头尾是“[]”的除外）。

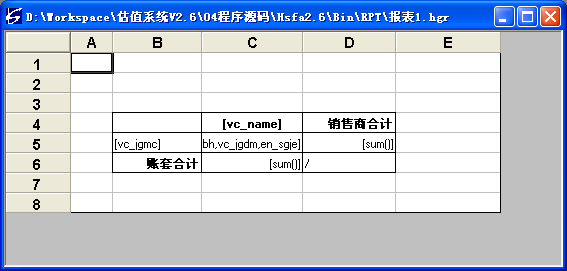
示例：统计基金账套的各销售商的申购金额

【配置结果】

配置的SQL脚本：

|  |
| --- |
| **select** t.l\_ztbh, s. vc\_name, t.vc\_jgdm, t.vc\_jgmc, t.en\_sgje **from** tsgsh t, tsysinfo s **where** t.l\_import=1 **and** s.l\_id=t.l\_ztbh **and** s.vc\_ztlx='1' **and** t.d\_qrrq=to\_date('20080606', 'yyyymmdd') **order** **by** t.l\_ztbh, t.vc\_jgdm |

配置的报表窗口：



运行效果：



#### 报表开发模板TReportListFrame

目前已提供了报表二次开发配置工具，一般的简单报表可以通过配置（存储过程、XML配置和报表模板开发）完成。二次开发请详见相关说明“[报表的二次开发和配置](#_报表的二次开发和配置)”。

使用报表开发模板TReportListFrame可以简便的开发各种类Excel报表。报表界面功能有特殊化的（如模板根据账套切换、DBF导出等）需要使用开发模板进行个性化开发。

【相关资源】

报表模块：hsReport.dll

开发模板位置：hsReport\hsReportListFrame.pas

报表控件位置：Control\Developer Express Inc\ExpressSpreadSheet 1.25

【开发方法】

1. 新建窗体，并放置一个向上靠齐的Panel作为参数栏。
2. 放置开发模板TReportListFrame，并在窗体显示事件OnShow中加入前面创建的参数栏：  
   Rpt.CoolBar1.Bands.Add.Control := Panel\_Param; //注：Rpt为报表模板
3. 根据需要放置参数，并在窗体创建事件OnCreate中初始化参数栏中的参数。同时放置『查询』按钮，并在按钮点击事件中调用数据引擎的对应查询方法。

【示例】

详见估值表（模块入口：hsReport.dll|ShowReportForm|估值表）。

#### 报表界面规范

1、字体

大标题 黑体 20号 粗体 合并单元格居中

小标题 黑体 10号 粗体 合并单元格居中

表格标题 宋体 10号 粗体 居中

表格文本 宋体 11号 居左

表格数值 宋体 11号 居右

对数据量特别大的报表如估值表，表格的字体可以酌情缩减一号。

整个报表要求最小字体10号，数值列宽足够打印。

单元格的高度根据字体设置，不能设置为小于字体所需高度。

2、内容格式

大标题为 “ @TITLE ”

小标题格式为“ @CorpName\_\_\_@FundName\_\_\_专用表 ”

显示表头日期参数，单个日期显示“ 日期：@Date ”，日期段显示“ 日期范围：@SDate ～ @EDate ”

3、颜色

所有的字体颜色为黑色

表格标题行背景色为 浅黄色 （标准色黄色的下面那个）。

4、线条

表格要求有1号黑色的实线填充

整个报表不显示灰色单元格线条

如果报表数据较多（正常屏幕需要向下滚动），需要增加冻结行（冻结线设置在表格标题下方）。

5、打印页面设置

页面边距全部设置为 15 mm

页面打印方向一般设置为纵向；如果打印预览时页面宽度超出页宽、且横向打印可以将当前页打印完整，则设置为横向。

6、合并单元格

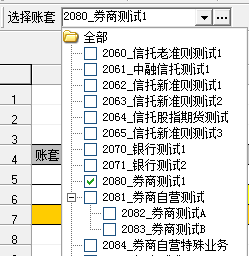
标题合并单元格 表格宽度超过页面宽度时，标题宽度设置为略小于页面宽度，使得标题在当前页显示居中

除多表头时外，表格标题尽量避免使用合并单元格。

## 其它说明

### 常用三方控件使用说明（推荐）

#### 账套控件



账套控件的几个属性和方法：

1）单选多选

    MutiSelect=false，单选。 单选时可启用ShowAllSelect属性，该属性起到了单选+选择全部的功能

    MutiSelect=true，多选。

注意1：单选+选择全部，或者多选时，Value属性值是以“,”分割的账套列表形式存在的，sql语句里面需要用in语法

注意2：保险版账套数可能会大于1000，在多选或者单选+全部模式时，该参数在定义xml配置时，必须设置Single=“false”

2）BookType属性，设置账套类型，下列类型可选：

* 用户账套，基金账套+分组账套+专户账套

一般维护数据时使用，如银行间债券交易

* 基金账套，基金账套+专户账套

一般设置基金层面配置时使用，如设置业绩报酬

* 实体账套，基金账套+信托分组账套
* 管理账套

基金账套+专户管理账套（不包括专户账套），设置科目相关参数使用，如科目设置、银行存款计息设置等

* 统计账套

统计账套+专户管理账套，一般报表查询使用，跨账套统计报表

* 统计账套+基金账套

统计账套+专户管理账套+基金账套，一般报表查询使用，大部分报表使用该设置

* 分组账套

一般报表查询使用，仅统计分组的报表，包括普通分组和信托分组

* 基金账套+账套组

查询行情数据时使用

3）账套过滤显示

    账套控件已经根据登陆账套的类型、当前模块的权限、给定EntryID 过滤账套列表

4）权限

    Grant       当前选择的所有账套的权限合集

    GrantAll    当前模块可用账套的权限集合，与选择无关，相当于全选账套的权限

    GetBookGrant(BookID)  当前模块中，指定账套的权限

5）配置方式实现报表，在配置报表xml时，需要设置BookType

    未设置BookType，则默认为“统计账套+基金账套”

6）账套值Value

账套值为账套编号或者账套编号的列表，如：“1002”、“1002,1003”

全选时，界面显示全部，值Value为所有可选的账套编号列表

界面打开时，单选模式下，默认选择的账套为登陆账套；如果登陆账套不在可选列表中，则默认默认相关联的账套；关联账套也无法选择，则默认选择第一个账套。多选模式下，默认选择的账套为登陆账套。

可对账套控件赋值。

7）ShowAllSelect属性，在单选时有效，即在单选时可选择全部

多选或者单选全部模式下，调用数据服务的xml配置中，需要注意账套数大于1000的情景。

#### 交易对手



交易对手控件可直接赋值为对手编号，显示对手名称。

属性CType，下列可选项：

交易对手

投资客户

受益人

发行机构

代持方

金融机构 等交易对手类型的分类

属性Value，交易对手编号

#### 下拉框ThsComboBoxValue（Hundsun Lib）

该控件具备了数值（Value）和文本显示（Text）的功能，即输入框显示文本，Value取值为对应的数值。

* 注意：

一般的，使用下拉框控件时要设置Style = csDropDownList，以确保只能够选择，而不能任意输入。

下拉框的下拉个数设置为16个或者更多（24、32）。

【示例1】下拉框（下拉框使用市场编号，显示市场名称）

|  |
| --- |
| //初始化交易市场  if FillDictionaryToItems('交易市场', boxMarket.Values, boxMarket.Items)>0 then  boxMarket.ItemIndex := 0; |

如上面的例子中，如果设置了（boxMarket.Value:=’1’;）则显示“上交所”；如果下拉选择了“上交所”，则boxMarket.Value输出“1”。

【示例2】下拉框，支持根据输入内容筛选下拉列表的功能

|  |
| --- |
| Box1.MatchValues.Add('1958'); Box1.MatchItems.Add('600570 恒生电子');  Box1.MatchValues.Add('452'); Box1.MatchItems.Add('600571 信雅达 ');  Box1.MatchValues.Add('2027'); Box1.MatchItems.Add('600572 康恩贝 ');  Box1.MatchValues.Add('453'); Box1.MatchItems.Add('600573 惠泉啤酒');  Box1.MatchValues.Add('454'); Box1.MatchItems.Add('600575 芜湖港 ');  Box1.MatchValues.Add('455'); Box1.MatchItems.Add('600576 万好万家');  Box1.MatchText := true; //根据输入项匹配文本值，并在下拉列表中显示匹配后的内容  //Box1.MatchValue := true; //根据输入项匹配数值，并在下拉列表中显示匹配后的内容 |

#### 列表框ThsListBoxValue（Hundsun Lib3）

同ThsComboBoxValue相似，在TListBox基础上增加了数值（Value）功能。

#### 带按钮的编辑框ComboEdit（RX Library）

#### 带计算器的输入框TRxCalcEdit（RX Library）

#### 数值和金额输入框TCurrencyEdit（RX Library）

【整数输入】

FormatOnEditing := true;

DecimalPlaces := 0;

DisplayFormat := ',0';

【金额输入】

FormatOnEditing := true;

DisplayFormat := ‘,0.00’;

DecimalPlaces := 2;

【比例输入】

FormatOnEditing := true;

DisplayFormat := ‘,0.00####’;

DecimalPlaces := 6;

注：根据业务的有效位数确认，最小2位。

注意：该组件在赋值和取值时用Value属性（property Value: Extended）

#### 表格控件TDBGridEh（EhLib）

【优点1：编码字段自动显示文本】

TDBGridEh的列具备了关联Dataset编号字段，在列中显示对应文本的功能。

优点：不需要在Dataset中编码改写GetText，也不需要在Grid中改写列文本显示方法。

示例：表格列值显示为名称（市场编号显示为市场名称）

|  |
| --- |
| //初始化表格列，使表格TDBGridEh的市场编号列，显示为市场名称。这里第4列为市场编号  procedure TAccountMngForm.FormCreate(Sender: TObject);  function GetColEh(sField: String): TColumnEh;  var i: Integer;  begin  for i:=0 to AccountList.DBGridEh1.Columns.Count-1 do  begin  Result := AccountList.DBGridEh1.Columns[i];  if SameText(Result.FieldName, sField) then  Exit;  end;  Result := nil;  end;  var  colEh: TColumnEh;  begin  colEh := GetColEh('l\_sclb');  if Assigned(colEh) then  FillDictionaryToItems('交易市场', colEh.KeyList, colEh.PickList);  end; |

【数字型字段的格式化显示】

|  |
| --- |
| //在数据集TDataset打开后设置字段格式  ……  DataSet1.Recordset := rs;  TNumericField(DataSet1.FindField('en\_je')).DisplayFormat := ',0.00';  TNumericField(DataSet1.FindField('l\_sl')).DisplayFormat := ',0';  …… |

#### 多选下拉框hsCheckComboBox

该控件具备了数值（Value）和文本显示（Text）的功能，即下拉框输入框显示文本列表，Value取值为选中的数值列表（如“1,2,3”），输入框可以显示选中的数值列表或者文本列表。

* 注意：

一般的，使用下拉框控件时要设置Style = csDropDown，以确保多选的文本值能够显示在输入框中。

下拉框的下拉个数设置为16个或者更多（24、32）。

【示例1】下拉框（下拉框使用市场编号，显示市场名称）

|  |
| --- |
| //初始化交易市场  if FillDictionaryToItems('交易市场', boxMarket.Values, boxMarket.Items)>0 then  boxMarket.ItemIndex := 0; |

如上面的例子中：

* 1. 当ShowValue=False，如果设置了（boxMarket.Value:=’1,2’;）则显示“上交所,深交所”；如果下拉选择了“上交所”，则boxMarket.Value输出“1”；如果下拉选择了“上交所”、“深交所”两项，则boxMarket.Value输出“1,2”；
  2. 当ShowValue=True，如果设置了（boxMarket.Value:=’1,2’;）则显示“1,2”；如果下拉选择了“上交所”，则boxMarket.Value输出“1” ；如果下拉选择了“上交所”、“深交所”两项，则boxMarket.Value输出“1,2”。

#### 日历选择下拉框ThsfDateTimePicker

比TDateTimePicker增加了属性Value、事件OnDateTimeChange

property Value: Variant;

Value属性可接受各种数据类型的日期赋值，内部会转换为日期格式；日期模式下输出YYYYMMDD的字符串，时间模式下输出HH:NN:SS的字符串。

**特别的**，Value可输入0或者空值（null, 空字符串），使得控件显示为空日期。

Procedure OnDateTimeChange (Sender: TObject; OldDateTime, NewDateTime: TDateTime);

只在日期或者时间改变时触发OnDateTimeChange事件；有别于OnChange事件在日期框下拉未改变日期时也触发事件的问题。

推荐在业务编辑界面中日期联动功能中使用。

### 特殊功能的三方控件使用

#### ~~凭证控件TVoucherGrid（VoucherCtl）~~

#### 日历控件hsMonthCalendar

### 主程序框架的Debug调试模式

在两层模式下，支持数据服务的调试模式和自动更新模式

数据服务引擎配置文件：DataService.xml

根节点下设置“Debug="true"”，即打开数据引擎的调试模式。设置“AutoRefresh="true"”，即打开运行时即时更新模式，支持边修改配置文件边保存，保存后不用重启程序而能够直接失效。

主程序的Ini文件：HSValuate.ini

主程序下增加[System]项目，该项目下增加“Debug=true”，即打开主程序的调试模式。主程序调试模式打开后，在系统菜单下有“数据服务测试”菜单，该模块提供数据服务接口配置的开发测试窗口。

### 多账套查询（账套数大于1000）

【数据服务引擎的处理】

数据服务引擎中，对同时满足以下条件的，进行特殊处理：

1、xml中参数配置包含了 Single="false" 选项的；

2、参数值包含了以“,”分割的，个数超过1000个的；

3、数据库连接驱动为 OraOLEDB.Oracle.1

不满足以上任一个条件时，按原参数格式（考虑到兼容以前的报表可以正常运行）。

超过1000个的特殊处理方法：

1、对查询和执行的sql语句，对参数进行自动替换

select id from TMP\_IDVlaues

2、对存储过程参数替换为

[LIST]

3、替换后的参数，其数据进入了临时表TMP\_IDVlaues，仅对当前连接有效

create global temporary table TMP\_IDVlaues

(

id number

)

on commit preserve rows;

【自定义查询和存储过程SQL改造】

对存储过程实现的批量账套查询报表

假设原存储过程取数条件为：and instr(','||ztlist||',', ','||l\_id||',')

需要改造为：

and (instr(','||ztlist||',', ','||l\_id||',')>0) or (ztlist='[LIST]' and l\_id in (select id from TMP\_IDVlaues))

或者改造为：

and (instr(','||ztlist||',', ','||l\_id||',')>0) or (ztlist='[LIST]' and exists (select 1 from TMP\_IDVlaues t where t.id=l\_id))

## Q&A

### 数据集转换函数XmlToRecordSet异常

Q：数据集转换函数XmlToRecordSet异常信息：“对象已在集合中。无法追加。”



A：数据引擎配置问题，字段名称重复。

### 清算模块－数据导入异常

Q：清算模块的数据导入异常信息，日志显示为

|  |
| --- |
| 『账套』Paper测试基金  『业务』交易所行情--sjshq  『操作』接收  『执行结果』文件无法打开。可能文件已损坏或者被占用  未找到提供者。可能未被正确安装。。Code=-203  『操作时间』20080711 11:50:40 |

A：可能原因是

1. 未安装相关文件的驱动程序，如果是DBF文件，需要注册安装FoxPro OLEDB驱动；
2. 文件格式错误、损坏，或者被其它程序占用；

【提示】

安装FoxPro OLEDB驱动方法，在命令行执行：

regsvr32 “C:\Program Files\Hundsun\hsfa\vfpoledb.dll”

上面的路径“C:\Program Files\Hundsun\hsfa”为本系统的安装路径，请根据实际情况配置。

Q：清算模块的数据导入异常信息，日志显示为

|  |
| --- |
| 『账套』Paper测试基金  『业务』交易所行情--sjshq  『操作』接收  『执行结果』文件读取错误，格式不匹配。请检查文件格式是否正确  数据提供程序或其他服务返回 E\_FAIL 状态。Code=-204  『操作时间』20080711 11:50:40 |

A：可能原因是，DBF文件格式错误。如DBF文件的数据越界（字段值超出字段定义范围或者精度）。

### 内存管理

Q：Delphi下面的代码会有内存泄漏

|  |
| --- |
| type  Ptmp=^Rtmp;  Rtmp=record  sTemp: string;  end;    var p: Ptmp;  begin  new(p)  try  p^.sTemp := 'sfdfdfdkkkkkkkkkkkkkkkkkkk';  ……    finally  FreeMem(p);  end;  end; |

A：上面是错误的内存管理编码

Delphi提供了两组内存管理接口：

【New、Dispose】

带变量初始化、变量释放处理的内存管理接口，适用于结构体、指针的处理。

【GetMem、FreeMem】

纯内存管理接口，申请和释放内存，不处理内部的变量。

上面两组接口一般不能交叉使用，否则会引起内存泄漏。前面的例子就是将New与FreeMem交叉使用了。

在C++里面，内存管理也需要配对：

【new、delete】适用于指针（结构体、数组、对象等的指针）的处理

【malloc、free】直接对内存进行管理。

Q：C++下面的代码会有内存越界

|  |
| --- |
| {  char \* c = new char[16];       memset(c, 0, strlen(c));       IntToStr(operator int(), c, 10);       string s = c;       delete c;       return s;  } |

A：strlen是以\0为标志取长度的，可能小于或者大于申请的长度

当小于申请的长度时，未实现所预期的全部清零的目的；

当大于申请的长度时，内存清零越界了，导致其他的内容被清零（如对象、指针、数据、变量等）。

## 附

### FA3.0中间件通讯协议

客户端与中间件之间的数据通讯采用了T2协议的TAG\_MESSAGE\_BODY部分传输二进制数据。我们将自己的数据转换为**数据流**，然后写入到TAG\_MESSAGE\_BODY中，通过T2协议发送。

下面介绍的是FA3.0的如何将请求参数转换为**数据流**。

FA3.0的请求参数类型包括3类：整形、字符串、二进制。这些数据写入数据流的格式为：

【整形】直接写入4个字节。整形中包括指针。

【字符串】先写入字符串长度（整形），再写入字符串

【二进制】先写入二进制数据长度（整形），再写入二进制数据

【1001】通用业务服务BussinessMethod

请求数据包格式

TDataPkg指针

MethodID

sSession(xml: UserID/UserCode/UserName/SubSysID/AccountState)

sParam

返回数据包格式

TDataPkg指针

iReturn

sData

sError

【1002】通用业务服务BussinessMethodEx

请求数据包格式

TDataPkg指针

sConfig

sSession(xml: UserID/UserCode/UserName/SubSysID/AccountState)

sParam

返回数据包格式

TDataPkg指针

iReturn

sData

sError

【1003】其他数据服务BussinessStream

请求数据包格式

TDataPkg指针

sFiles(xml格式)

sSession(xml: UserID/UserCode/UserName/SubSysID/AccountState)

sParam

其中sFiles格式为：

<Files Count="" Dir=""><File Index="0">filename1</File><File Index="1">filename2</File></Files>

返回数据包格式

TDataPkg指针

iReturn

sData

sError

【1004】文件服务接口，包括获取上传批号、上传文件信息、下载文件

请求数据包格式

Data指针

sParam

0

返回数据包格式

TDataPkg指针

iReturn

sData

sError

【1005】文件服务子接口，分包上传文件

请求数据包格式

Data指针

sParam

bData二进制数据

返回数据包格式

TDataPkg指针

iReturn

sData

sError

【1006】文件服务子接口，分包下载文件，服务端推送数据

返回数据包格式

TDataPkg指针

iIndex=分包序号

分包个数

分包大小

bData二进制

其他数据服务BussinessStream的文件流数组streams如果不为空，则会内部调用文件上传服务，流程为：

获取上传批号、上传文件信息、分包上传文件。具体的程序控制流水描述：

1. 获取发送批次号，服务端准备好该批次x对应的唯一目录，获取应答耗时N

1004：sParam=<Dir/>

返回sData：批次号x

1. 逐个发送文件，一个文件成功发送后再发下一个

每个文件是分包发送的，服务端异步返回接收的分包序号

* 1. 发送文件名称、文件大小

1004：sParam= <AFile DName="x" FName="filename1" Size=""/>

返回sData：文件相对路径“/tmp/x/filename1”或者“\tmp\x\filename1”

* 1. 连续发送10个，等待5N，再发送下一批

1005：sParam=<PFile Name="/tmp/x/filename1" PKG\_Index="分包序号" PKG\_Size="16K"/>

bData：二进制文件分包

返回sData：空

返回iReturn：分包序号

* 1. 全部发送完成，最近接收回报超过10\*N后，重发没有应答的数据包，下次重发间隔再增加10\*N。重发8次，超时失败终止。

文件下载服务接口BussinessGetStream，是通过1004访问的

1. 给定文件名称（含相对路径），获取文件信息

1004：sParam=<GFile>filename</GFile>

返回sData：<File Size="文件大小" PKG\_Size="分包大小">filename</File>

1. 给定文件名称和分包序号列表，获取对应分包的二进制数据

请求1004：sParam=<GFile PKGS="0,1,2,3,4,5,6">filename</GFile>

1006返回每个请求分包的二进制数据

1004返回sData：<File Size="文件大小" PKG\_Size="分包大小">filename</File>