FScada.Net 高级编辑手册（一）

目录

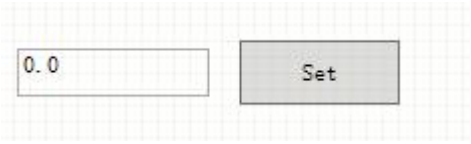
借助于代码感知技术在 FScada.net 中可以轻松编写 CSharp 代码：

```
public void button1_Click(object sender,RoutedEventArgs e)
{
    TextBox box = (TextBox)schemaCanvas.FindName("edit1");
    if (box!=null)
    {
        IChannel ch = Global.GetChannel("sim.v1");
        //ch.Value = 1.0f;
        float f = 0.0f;
        if (float.TryParse(box.Text,out f))
        {
            ch.Value = f;
        }
        else
           Global.MyApp.ErrorMsgBox("设置失败！","Error");
        Global.
    }
}

public void butt                                    utedEventArgs e)
{
    Button butto
    button.Conte                                    String();
}

public void Canv                                    tedEventArgs e)
{
    Button button = (Button)schemaCanvas.FindName("button2");
    button.IsEnabled = false;
}
```

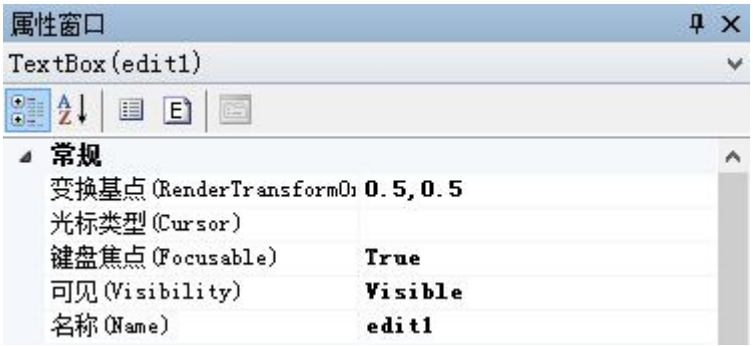
Equals
GetChannel
MyApp
MyLogger
MyProject
OnProjectStart
OnProjectStop
ReferenceEquals

更可以通过“项目”菜单导出到 VS 功能，把项目文件导出到 VS2010 中进行代码开发。

1、编程读取 TextBox 控件



设置 TextBox 控件名称为“edit1”，按钮的 Click 事件“button1_Click”



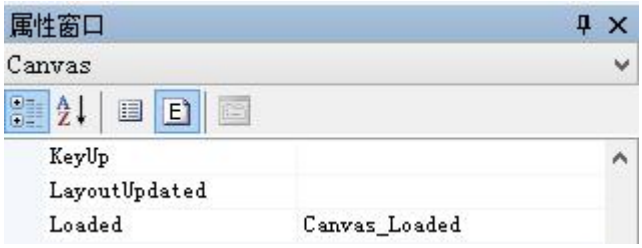
通过按钮点击获取输入内容设置标签值的参考代码如下（sim.v1 为一可读写的 Single 类型变量）：

```
public void button1_Click(object sender,RoutedEventArgs e)
```

```
{
    TextBox box = (TextBox)schemaCanvas.FindName("edit1");
    if (box!=null)
    {
      IChannel ch = Global.GetChannel("sim.v1");
      //ch.Value = 1.0f;
      float f = 0.0f;
      if (float.TryParse(box.Text,out f))
      {
        ch.Value = f;
      }
      else
        Global.MyApp.ErrorMsgBox("设置失败！","Error");
    }
}
```

2、页面载入时执行函数

例：页面载入时设置按钮




```
public void Canvas_Loaded(object sender,RoutedEventArgs e)
{
    Button button = (Button)schemaCanvas.FindName("button2");
    button.IsEnabled = false;
}
```

3、控件的初始化操作

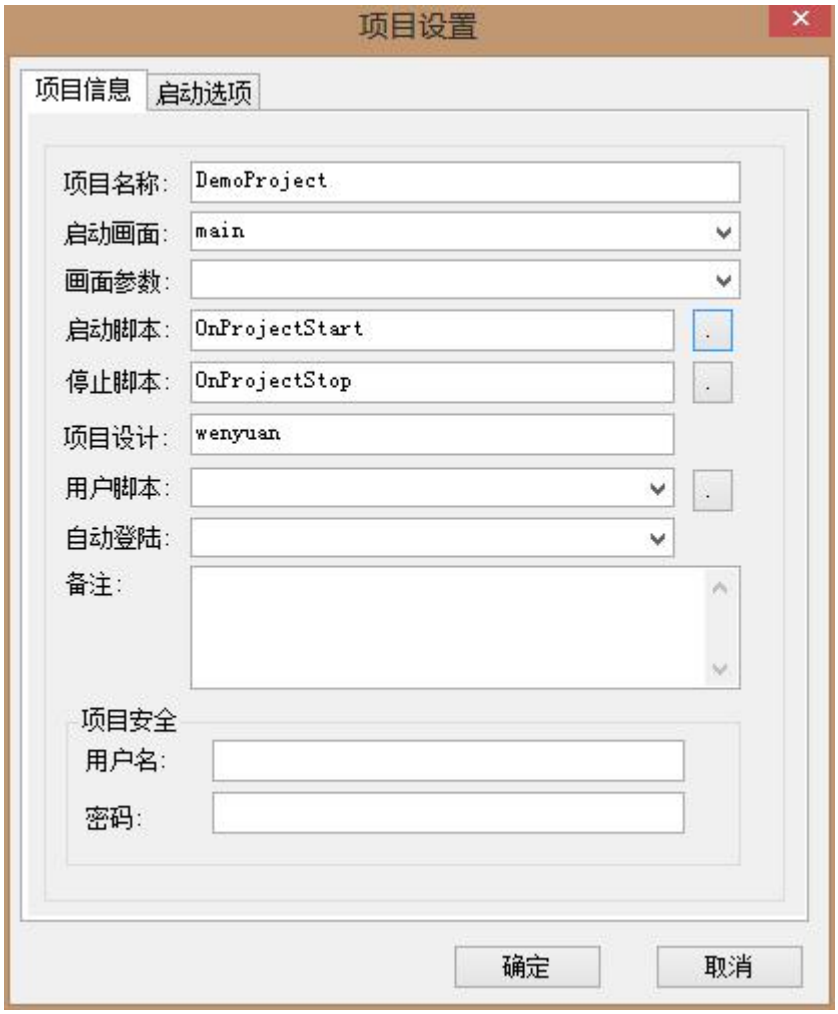
例：按钮初始化时设置显示内容


```
public void button2_Loaded(object sender,RoutedEventArgs e)
```

```
{
    Button button = sender as Button;
    button.Content = System.DateTime.Now.ToString();
}
```

4、项目启动停止时执行函数

项目启动停止执行函数必须写在 global 文件中，函数必须为静态无参数、无返回的函数。



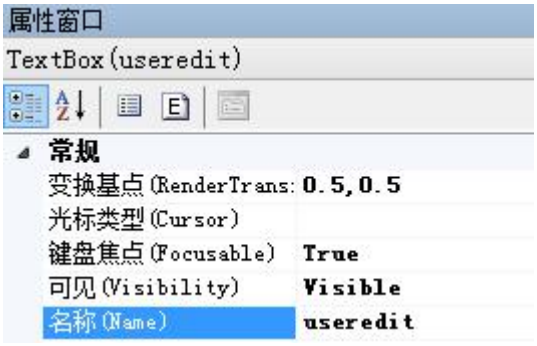
例如（如果在启动选项里面找不到函数，先编译下代码）：

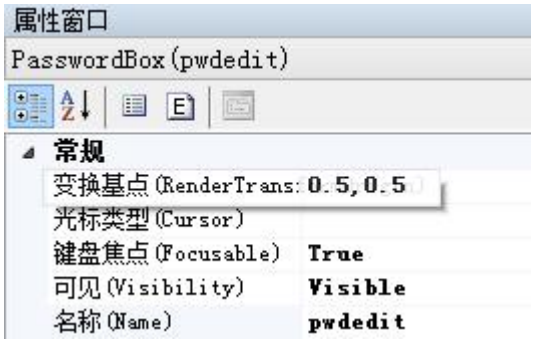
```
//<summary>
//项目启动执行函数
//</summary>
public static void OnProjectStart()
{
}

//<summary>
//项目停止执行函数
```

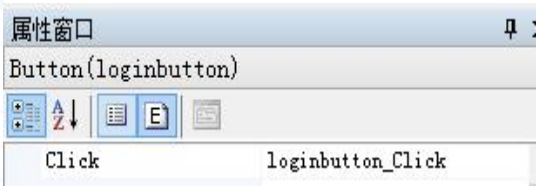
```
//</summary>
public static void OnProjectStop()
{
}
```

5、自定义登陆实现方法

文本输入控件 useredit、密码输入控件 pwdedit、按钮控件 loginbutton



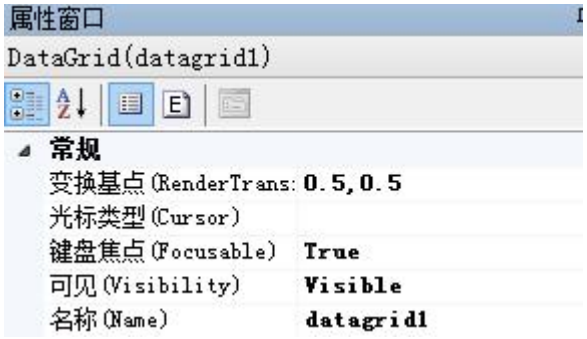


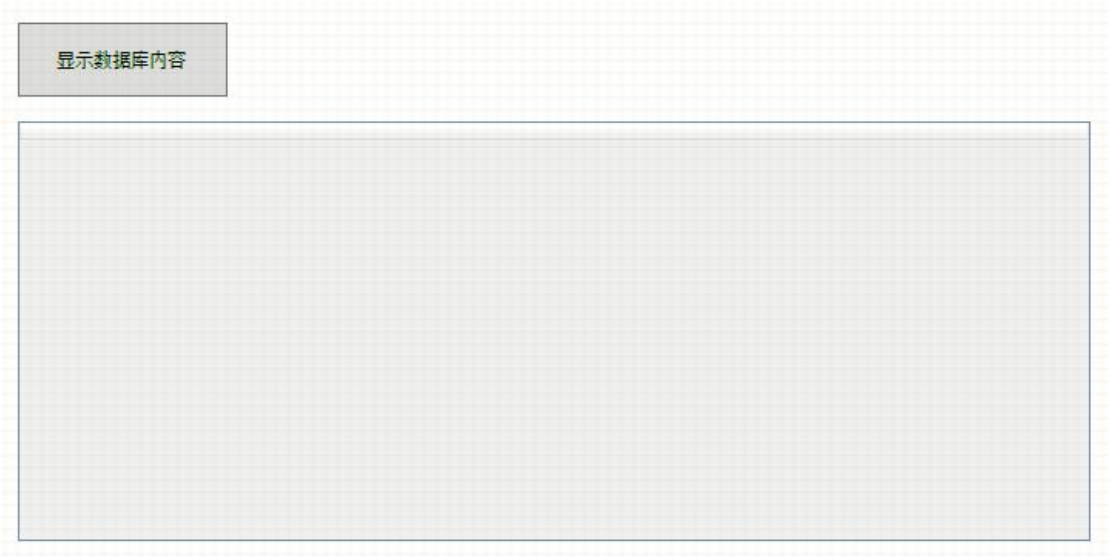


在按钮的 Click 事件编写登陆代码

```
public void loginbutton_Click(object sender,RoutedEventArgs e)
{
   TextBox userbox = (TextBox)schemaCanvas.FindName("useredit");
   PasswordBox pwdbox = (PasswordBox)schemaCanvas.FindName("pwdedit");
   if (Global.MyApp.Login(userbox.Text,pwdbox.Password))
   {//成功登陆
   }
   else//登陆失败
   {
   }
}
```

6、从数据库读取数据显示到画面





页面按钮代码：

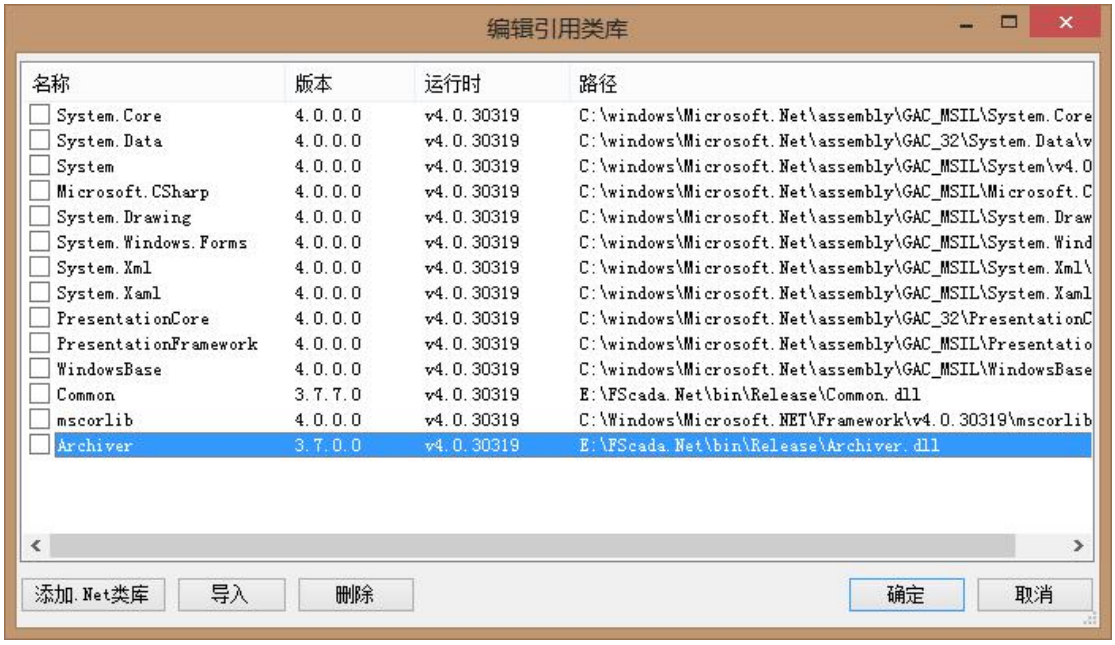
```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using FScada.Common;
using FScada.Interfaces;
using System.Reflection;
using System.Windows.Input;
using System.Data;
using System.Data.SqlClient;

namespace RunTime
{
    /// <summary>
    /// 画面脚本响应文件
    /// </summary>
    public  class Schema_datagrid : IDisposable
    {
        Canvas schemaCanvas = null;
```

```
//FrameworkElement
UIElement[] uiArray = null;

public Schema_datagrid(Canvas c,UIElement[] ar)
{
   schemaCanvas = c;
   uiArray = ar;
}

~Schema_datagrid()
{
}

  /// <summary>
 /// FScada 显式调用，用于释放资源
/// </summary>
public void Dispose()
{
   schemaCanvas = null;
}

string GetSqlConnection()
{
   SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();
   builder.Add("Data Source", ".\\SqlExpress");
   builder.Add("Integrated Security", "True");//使用信任权限
   // builder.Add("User Id", DbUser);
   // builder.Add("Password", DbPassword);
   builder.Add("Database", "fscada");//数据库名称
   return builder.ConnectionString;
}

public void button_Click(object sender,RoutedEventArgs e)//显示数据到控件
{
   DataGrid grid = (DataGrid)schemaCanvas.FindName("datagrid1");
   grid.AutoGenerateColumns = true;
   using(SqlConnection conn = new SqlConnection(GetSqlConnection()))
   {
       conn.Open();
       SqlDataAdapter dataadp = new SqlDataAdapter("Select * from [realTable]",conn);
       DataTable dt = new DataTable();
       dataadp.Fill(dt);
       grid.ItemsSource = dt.DefaultView;
   }
```

```
        }

    }
}
```

7、从代码中读取归档数据

在代码中添加 Archiver 引用



页面代码如下：

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using FScada.Common;
using FScada.Interfaces;
using System.Reflection;
using System.Windows.Input;
using FScada.Archiver;
using System.Data;
using System.Collections.Generic;

namespace RunTime
{
    /// <summary>
    /// 画面脚本响应文件
    /// </summary>
    public  class Schema_archiver : IDisposable
    {
        Canvas schemaCanvas = null;
```

```
        UIElement[] uiArray = null;

        public Schema_archiver(Canvas c,UIElement[] ar)
        {
            schemaCanvas = c;
            uiArray = ar;
        }

        ~Schema_archiver()
        {
        }

         /// <summary>
         /// FScada 显式调用，用于释放资源
        /// </summary>
        public void Dispose()
        {
            schemaCanvas = null;
        }


        //读取标签历史数据
        public void button1_Click(object sender,RoutedEventArgs e)
        {
            if (HisArchiver.Current.DatabaseSettings.EnableArchiving)
            {
                DateTime startTime = DateTime.Now.AddHours(-1);
                DateTime endTime = DateTime.Now;
                DataTable dt =  ClArchiver.Current. QueryData (startTime,endTime,"system.second");
                if (dt!=null)
                {
                }
            }
        }
    }
}
```

8、执行登录通知和登录判断

下面的代码写在 global.cs 中

```
        //<summary>
        //项目启动执行函数
        //</summary>
        public static void OnProjectStart()
```

```
{
  Env.Current.Project.UserLogin += new DelegateUserLogin(OnLogin);
}

//登录消息通知
private static void OnLogin(UserInfo u)
{
     if (u.UserName!="")
      {
        MessageBox.Show("Login");
      }
      else
      {
        MessageBox.Show("Logout");
      }
}

//<summary>
//项目停止执行函数
//</summary>
public static void OnProjectStop()
{
   Env.Current.Project.UserLogin -= new DelegateUserLogin(OnLogin);
}
```

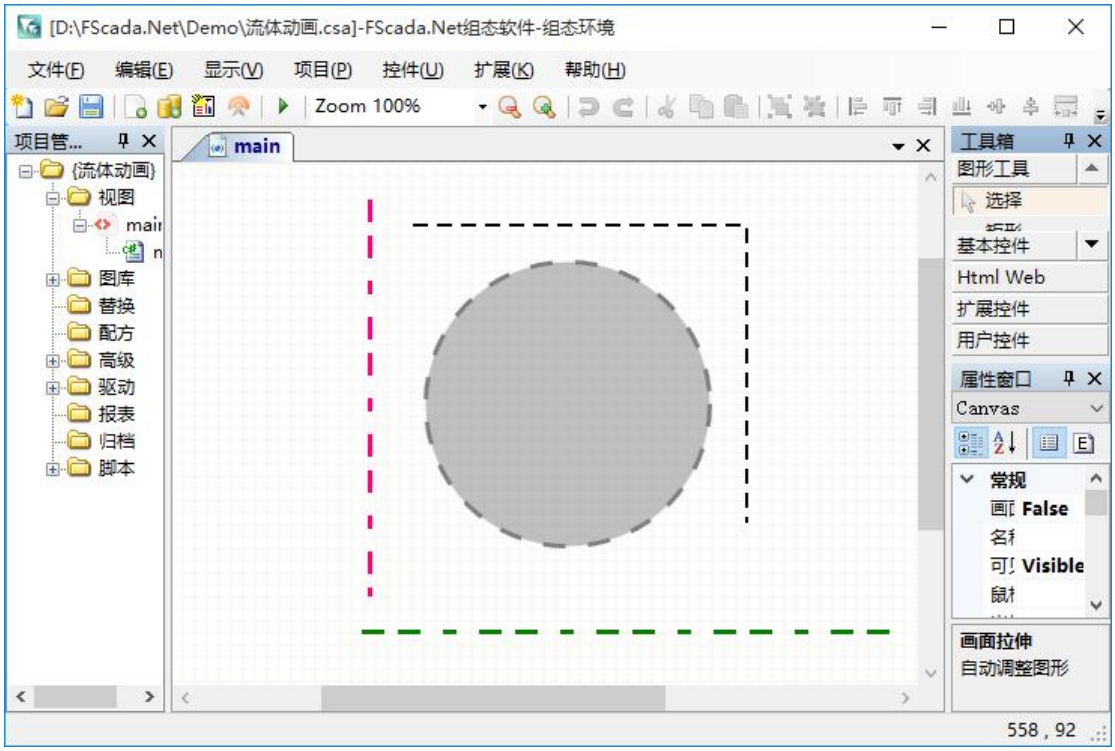
设置项目启动和停止函数后，用户发生登录和注销时就会收到消息

```
//下面 2 个函数启用自定义登录判断和注销

//用户登录执行函数，返回一个有效的 UserInfo 信息
public static UserInfo DoLogin(string username,string md5pwd)
{
   Env.Current.Logger.LogInfo("RunTime","Call User logoin");
   UserInfo user = null;
   user = new UserInfo("admin",100,0);
   return user;
}


//用户注销执行函数
public static void DoLogoff(string username)
{
   Env.Current.Logger.LogInfo("RunTime","Call User logoff");
}
```

如果检测到包括上面的代码，则用户判断登录有效。

9、流体流动效果

演示项目：流体动画.csa

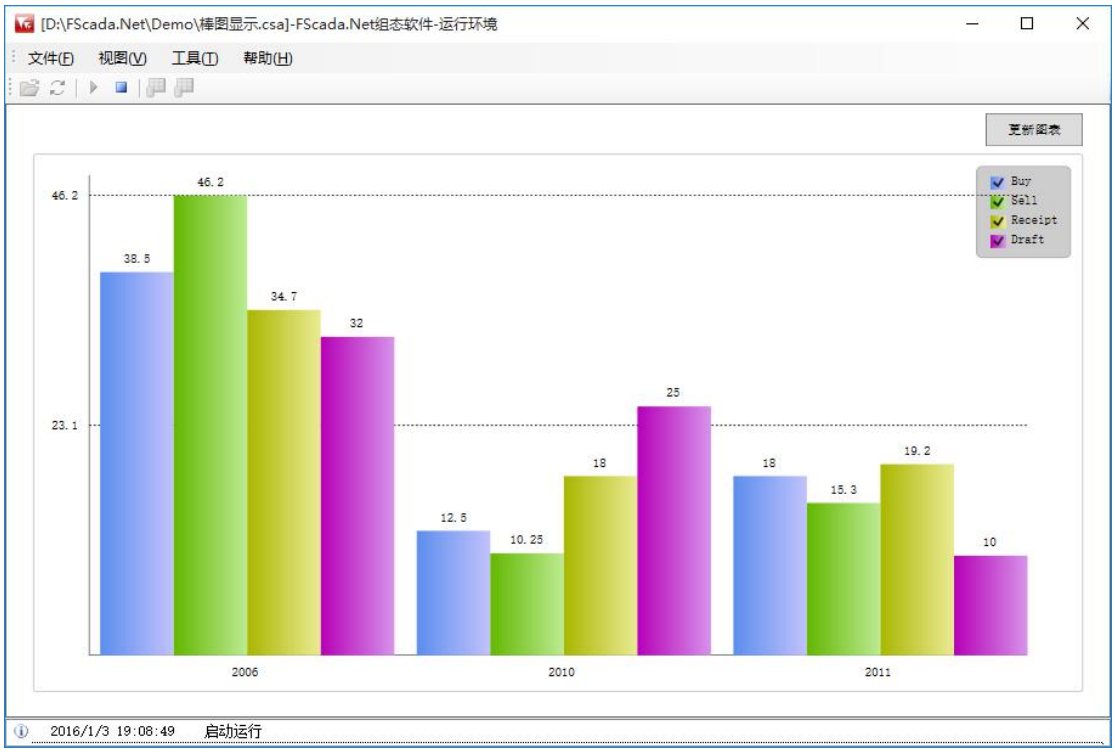


更改线条为虚线类型

改变“线冒偏移”就会产生流动效果

10、使用棒图

演示项目：棒图显示.csa
完全 C#代码实现，控件位于 FScada.Controls 命名空间下的 BarChart



11、动画效果显示

演示项目:WPF 动画.csa
完全 C#代码实现

12、替换系统登录对话框

```
//<summary>
//项目启动执行函数
//</summary>
public static void OnProjectStart()
{
    Env.Current.Application.OnLogin += OnFingerLogin;
}

//<summary>
//项目停止执行函数
//</summary>
public static void OnProjectStop()
{
    Env.Current.Application.OnLogin -= OnFingerLogin;
}

private static void OnFingerLogin(object sender,EventArgs e)
{
     if (sender.Equals("login")){
        FScada.LoginDialog dlg = new FScada.LoginDialog();
        if (dlg.ShowDialog() == DialogResult.OK)
        ...
       dlg.Dispose();
       dlg = null;
     }
     else
     {
         Logger.LogData log = new Logger.LogData();
         log.datetime = DateTime.Now;
         log.source="UserName";
         log.strdata = "注销";
         log.username = "";
         FScada.Archiver.LogArchiver.Archiver.Current.AddOperatorLog(log);
     }
}
```

13、自定义报警转向

```
//<summary>
//项目启动执行函数
//</summary>
public static void OnProjectStart()
{
    Env.Current.AlarmManager.OnGotoAlarm += OnGotoAlarm;
}

//<summary>
//项目停止执行函数
//</summary>
public static void OnProjectStop()
{
    Env.Current.AlarmManager.OnGotoAlarm -= OnGotoAlarm;
}


private static bool  OnGotoAlarm(BaseChannel channel,string tag)
{
    //tag:变量定义中的报警画面
    //channel.AlarmPic和tag意义相同
    return true;
    //返回 false执行系统默认操作
}
```

14、自定义 CSV 存储

```
//<summary>
//项目启动执行函数
//</summary>
public static void OnProjectStart()
{
    recRun = true;//设置启动标志
    ThreadPool.QueueUserWorkItem(RecThread,null); //启动线程
}

//<summary>
//项目停止执行函数
//</summary>
public static void OnProjectStop()
{
    recRun = false; //停止线程
}
```

```
public static void RecThread(object state)
{
    Env.Current.Logger.LogInfo("CSV Logger","Started");
    string filename = "rec.csv";
    filename = Path.Combine(Env.Current.StrLogFilePath, filename);
    try{
    Stream stream = new FileStream(filename, FileMode.OpenOrCreate, FileAccess.Write, FileShare.Read);
    if (stream != null){
        TextWriter writer = new StreamWriter(stream);
        stream.Seek(0,SeekOrigin.End);
        BaseChannel tag1 = Env.Current.GetBaseChannel("system.second");
        BaseChannel tag2 = Env.Current.GetBaseChannel("system.hour");
        if (tag1!=null && tag2!=null){
            if (writer != null){
            Stopwatch sp = new Stopwatch();
            int count = 0;
            while (recRun){
                sp.Restart();
                writer.WriteLine(string.Format("{0:yyyy-MM-dd HH:mm:ss.fff},{1},{2}", DateTime.Now,tag1.strValue,tag2.strValue));
                count ++;
                if (count > 10){ //5秒刷新磁盘
                   writer.Flush();
                   count =0;
                }
                if (sp.ElapsedMilliseconds < 500)
                {
                    Thread.Sleep(500 - (int)sp.ElapsedMilliseconds);
                }
            }
            writer.Close();
            stream.Close();
            }
        }
    }
    }
    catch (Exception ex)
    {
       Env.Current.Logger.LogError("CSV Logger",ex.Message);
    }
}
```

演示项目：存储 CSV.csa

15、GPS 通讯

使用 COM 口接收并解析 GPGGA 消息（GPS 定时发送），记录 GPS 坐标并且转换为百度地图坐标，接线 GPS 时间

```
static SerialPort port = null;
static WebClient wb = null;
static JavaScriptSerializer jar = null;

//<summary>
//项目启动执行函数
//</summary>
public static void OnProjectStart()
...


//转换为百度坐标
private static void ConvertPoint(double x, double y)
...

//<summary>
//项目停止执行函数
//</summary>
public static void OnProjectStop()
...
```

演示项目：GPS.csa

16、自定义 WinForm 报警控件

```
public void alcontrol_Loaded(object sender,RoutedEventArgs e)
{
     WinFormAlarmControl control = sender as WinFormAlarmControl;
     if (control!=null){
          control.AutoResizeCol = false;
          System.Windows.Forms.ListView listview = control.Control;
          if (listview!=null){
             listview.Columns[0].Text = "报警内容";
             listview.Columns[0].Width = 300;
          }
     }
}
```

17、自定义 WPF 报警控件

```
  public void WPFAlarmControl_Loaded1(object sender,RoutedEventArgs e)
  {
      WPFAlarmControl   ctl = sender as WPFAlarmControl;
      GridView gv =  ctl.Control.View  as GridView;
      GridViewColumn col1 =  gv.Columns[0];
      col1.Width = 120;
  }
```

18、计算某历史存储的最大值、最小值、平均值

```
public static void CaclValue(object state)
{
     DateTime end = DateTime.Now;
     DateTime start = end.AddHours(-1);
     double maxvalue=0,minvalue=0,avgvalue=0;
     try{
     if (Archiver.Current.GetValue(start,end,"sim_var_6",ref maxvalue,ref minvalue,ref avgvalue))
     {
        Global.MyApp.SetTagValue("max",maxvalue);
        Global.MyApp.SetTagValue("min",minvalue);
        Global.MyApp.SetTagValue("avg",avgvalue);
     }
     }
     catch
     {
     }
}
```

可以使用变量或者模拟驱动触发执行
Archiver.Current.GetMaxValue：查询最大值
Archiver.Current.GetMinValue：查询最小值

19、Excel 操作

```
public void Button_Click(object sender,RoutedEventArgs e)
{
     Excel excel = new Excel(); //创建Excel对象
    if (excel.LoadFromResource("realdata.xls")) //从项目中装载Excel文件 LoadFromFile
    {
        excel.SetCurrentSheet(0);                //设置当前工作表,后续操作在此工作表进行
        for (int i=1;i<28;i++){
            string tagname =  excel.GetCellStringValue(i,1); //从工资表指定行列读取标签名称
            if (!string.IsNullOrEmpty(tagname))
            {
               IChannel ch = Env.Current.GetChannel(tagname);
               if (ch!=null)
                   excel.SetCellValue(i,2,ch.Value);               //设置数据
            }
            //如果Sheet内有公式 则加下列代码 强制更新计算
            excel.CurrentSheetForceFormulaRecalculation = true;
        }
        //excel.SetActiveSheet(0);  //保存时可以设置活动工作表,打开Excel显示的活动工作表
        string filename = System.IO.Path.Combine(Env.Current.Application.BasePath,"Reports","realdata.xls");
        excel.SaveToFile(filename);//Save to file
        //Excel.OpenExcel(filename,"print");  //Print Excel
        Excel.OpenExcel(filename);  //openExcel
    }
}
```

填充 Excel 文件，保存到 Report 目录，并打开文件，适合制作 Excel 报表（利用内置 Report 存储，读取数据呈现到 Excel）

20、FScada 内置函数说明

string BasePath　　获得应用程序目录

string ProjectPath　　获得项目文件目录

void PlusTag(string tagName, string inteval)　　按指定的时间间隔发脉冲 inteval 单位毫秒

void SetTagValue(string tagName, string value)　设置标签值

void AddTagValue(string tagName, string value)　　标签值加上 value 值

void EditRecipes() 显示配方编辑器

void ToogleTagValue(string tagName) 切换标签值

void FullScreen(bool val)　全屏控制，true false

IChannel GetChannel(string name)　获取标签对象

void ReplaceSchema(string name, string param = "") 带参数（可选）替换画面

void ReplaceChild(string windowName, string childName, string schemaName, string param = "") 替换子画面

void PopupSchema(string name, string param = "")　显示弹出窗口

void CloseSchema(string name) 关闭画面

void OpenSchema(string name, string param = "")　打开画面

void OpenDialogSchema(string name, string param = "") 以模态对话框形式打开画面

void OpenVariables()　打开标签数据库浏览

Brush BrushFromColor(byte r, byte g, byte b)　rgb 转换为画刷

void ErrorMsgBox(string msg, string caption) 显示错误信息对话框

void InfoMsgBox(string msg, string caption)　显示消息对话框

bool QuestionMsgBox(string msg, string caption) 显示询问对话框

void Exit() 停止运行并退出软件

void ProjectStart() 启动运行

void ProjectStop() 停止运行

bool Login(string username, string password) 用户登录

void Login() 调用登录对话框

```
void Logout() 用户注销
void LoadRepice(string strfile) 装载配方
void OpenReportFromFile(string repName, string param,bool modal=false) 打开文件报表
void OpenReport(string repName, string param,bool modal=false) 打开内部报表
void ExtendSendCommand(string extName, string command) 发送扩展命令
bool SetValueDialog(string tagname) 显示标签设置对话框
bool SetStringValueDialog(string tagname) 显示字符标签设置对话框
bool SetNumberValueDialog(string tagname) 显示数值标签设置对话框
void WriteOperatorLog(string tagName, string value) 记录操作日志
void EditSchedule() 编辑定时调度
void ScreenKeyboard() 打开屏幕键盘
void Execute(string app, string param) 运行外部程序
void OpenIE(string url) 打开 IE 浏览器器 和指定的网址
void EditUser() 用户管理
void ChangeUserPassword()  修改用户密码
void ViewColArchiver()  显示历史查询窗口
void ExportReport(string reportname,string paramname,string typename) 导出报表（xml,
html,csv, rtf, xlsx,tif）
void PrintReport(string reportname, string paramname) 使用默认打印机打印报表
```

21、IOClient 通讯协议描述

1、IOClient 使用 TCP 协议和 FScada 软件进行 IO 数据通讯，实现服务器端到客户端的实时数据更新推送，和数据写入。

2、FScada 组态环境，设置项目菜单可以配置是否启用 IOClient 通讯功能，并配置 TCP 通讯端口，默认通讯端口是 9020，HTML5 版本在系统设置同样需要配置好服务器 io 地址和 tcp 通讯端口。

3、IOClient 通讯首先需要经过用户名和密码认证，用户名和密码在 FScada 组态环境用户管理进行设置。

4、客户端发往服务器的数据统一采用 json 格式，格式如下定义：

```
class IOServerPack
{
    public string packtype = "";
    public string param1 = "";
    public string param2 = "";
    public string param3 = "";
    public string param4 = "";
    public string param5 = "";
    public int paramcount = 0;
```

```
}
```

通过指定 packtype 描述功能请求

TCP 连接建立以后发生登录包：

```
IOServerPack pack = new IOServerPack();
    pack.packtype = "login";
    pack.paramcount = 3;
    pack.param1 = appstate["ioserveruser"].ToString();
    pack.param2 = md5.encrypt(appstate["ioserverpassword"].ToString());
    pack.param3 = "channels";
```

序列化成 json 字符串后通过 tcp 发送给服务器端

如果登录成功后读取第 1 个字节，如果是 1 则表明登录成功，如果是 0 登录失败

登录成功后立即可以开始接受数据，首先接收到的是标签列表数据，后续服务器发来的数据均有 10 个字节的包头，包头格式如下：

0xFF　4 个字节 4 个字节 0xFF　4 个字节为尾随数据长度

尾随数据是 xml 格式文本，使用 GZip 进行了压缩

标签列表 xml 文本格式如下：

```
<channels>
<channel name=”” plugid=”” level=”” rangemax=”” rangemin=”” unit=”” readony=”” type=”” deadzone=”” desc=”” digcount=”” store=”” area=”” />
...
</channels>
```

Name:标签名称 字符类型

Plugid：驱动名称 字符类型

Level：权限值 整数

Rangemax：量程上限 浮点数

Rangemix：量程下限 浮点数

Unit：单位 字符

Readonly：只读 布尔

Type：数据类型 字符

Deadzone：死区 浮点数

Desc：描述 文本

Digcount：小数点个数 整数

Store：历史归档标志 布尔

Area：标签分组区 整数

标签列表接收完成后，服务器开始推送带 10 个字节包头的 GZIP 压缩后的 io 数据，数据格式是 xml 文本，格式定义如下：

```
<tagvalues>
<tagvalue id=" " v=" " s=" " a=" " t=" " />
...
</channels>
```

Id：标签名称

V：数值 布尔为 True 或 False

S：状态 整数 1：GOOD

T：日期时间

5、IO 数据控制数据包格式

```
class IOSetValuePack
{
    public string command;
    public string tagname;
    public string strvalue;
}
```

Command：命令类型 write toogle add

Tagname：标签名称

Strvalue：数据文本

写入 IO 值的代码描述如下：

```
IOServerPack pack = new IOServerPack();
   pack.packtype = "write";
   pack.paramcount = 1;
   List<IOSetValuePack> values = new List<IOSetValuePack>();
          for (int i = 0; i < writeChannels.Count; i++)
          {
                 IOSetValuePack v = new IOSetValuePack()
                 {
                                command = writeChannels[i].Command,
                                tagname = writeChannels[i].Name,
                                strvalue = writeChannels[i].Value
```

```
            };
                        values.Add(v);
        }
    if (values.Count > 0)
    {
        pack.param1 = jar.Serialize(values.ToArray());
    }
```

把 pack 序列化为 json 字符通过 socket 发送给服务器就完成了数据写入。

实现代码：Server/Scada/IOClient.cs

22、归档数据查询接口协议描述

1、数据查询接口使用 TCP 协议和 FScada 软件进行通讯，默认通信端口 125。

2、客户端发往服务器的数据统一采用 json 格式，格式如下定义：

```
class DataService_ReceiveData
{
    public string username = "";
    public string pwd = "";
    public string command = "";
    public string param1 = "";
    public string param2 = "";
    public string param3 = "";
    public string param4 = "";
    public string param5 = "";
    public string param6 = "";
    public int paramcount = 0;
}
```

通过指定 command 描述功能请求

1）查询报表归档数据

```
ReadReportData(string table,string tag,DateTime start,DateTime end, int second)
 DataService_ReceiveData send = new DataService_ReceiveData();
 send.command = "report_query";
 send.username = username;
 send.pwd = password;
 send.param1 = table;//数据表名称
 send.param2 = tag;//标签名称使用逗号分隔
 send.param3 = start.ToString("yyyy-M-d HH:mm:ss");
 send.param4 = end.ToString("yyyy-M-d HH:mm:ss");
 send.param5 = second.ToString();//没有意义
```

```
send.param6 = "";//空白返回 DataTable 格式 json 返回 json 格式 string 返回 csv 文本格式
send.paramcount = 6;
```

返回的数据包格式：4 个字节的包头，后面是包头长度的经过 GZip 压缩后的数据

2）历史数据查询

```
ReadHisData(string tag, DateTime start, DateTime end, int second)
DataService_ReceiveData send = new DataService_ReceiveData();
send.command = "his_query";
send.username = username;
send.pwd = password;
send.paramcount = 5;
send.param1 = tag;//标签名称使用逗号分隔
send.param2 = start.ToString("yyyy-M-d HH:mm:ss");
send.param3 = end.ToString("yyyy-M-d HH:mm:ss");
send.param4 = second.ToString();//数据精度
send.param5 = "";//空白返回 DataTable 格式 json 返回 json 格式 string 返回 csv 文本格式
```

返回的数据包格式：4 个字节的包头，后面是包头长度的经过 GZip 压缩后的数据

3）查询最近的报警

```
DataService_ReceiveData send = new DataService_ReceiveData();
send.command = "realalarms";
send.username = username;
send.pwd = password;
send.paramcount = 0;
```

4）报警确认操纵

```
DataService_ReceiveData send = new DataService_ReceiveData();
send.command = "ackalarm";
send.username = username;
send.pwd = password;
send.paramcount = 1;
send.param1 = “1;2”;//报警 id 编号使用分号进行分割
```

5）查询历史报警数据

```
DataService_ReceiveData send = new DataService_ReceiveData();
send.command = "alarmlog_query";
send.username = username;
send.pwd = password;
send.paramcount = 3;
send.param1 = start.ToString("yyyy-M-d HH:mm:ss");
send.param2 = end.ToString("yyyy-M-d HH:mm:ss");
send.param3 = "";//空白返回 DataTable 格式 json 返回 json 格式 string 返回 csv 文本格式
```

常州文庭软件有限公司
2017.1