13 日志服务

在软件开发中，日志服务是提升系统健壮性相当重要的组件。有了它，系统维护人员可以跟踪系统的异常状况，而系统管理人员可以跟踪敏感数据的变动状况。

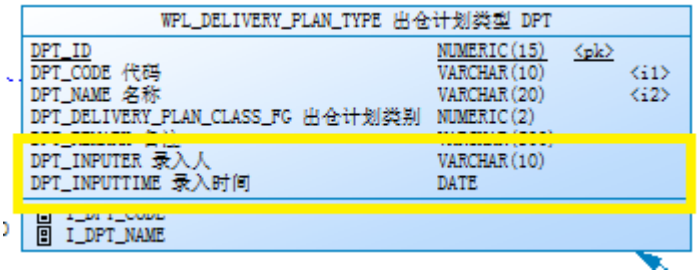
日志服务提供两种形式的记录日志方式，一种是由框架自动写日志，另一种是由业务对象主动控制写日志。日志主要用来记录用户操作历史、上机日志记录、某个用户在什么时间从哪台机器对哪个数据做了什么操作、是否成功执行等信息。

13.1 业务数据操作日志

应用系统中某些业务数据对于用户来说是蛮重要和关键的，其增删改等痕迹必须记录下来。由于这种记录方式具有普遍性，而且代码又是非常的繁琐无趣，那这种无聊的任务就交给 Phenix丶统一处理吧，但前提是要求开发人员按照一定规范进行设计。

13.1.1自动填充业务对象的录入信息

在业务数据上往往要记录当前的操作人和操作时间，而持久化时往往是记录在对应的表字段中，比如在表结构设计中会出现很多类似下图的表字段：



那么，只要我们统一命名了它们，剩下的工作（比如自动填充、持久化）就可以交给 Phenix丶来自动完成了。

为此，我们要遵守 Phenix丶在 Phenix.Core.Mapping.CodingStandards 类中约定的表字段命名规则：

属性	说明	备注
DefaultInputerColumnName	缺省“录入人工号”字段名	缺省值：字段名后缀为“_INPUTER”
DefaultInputTimeColumnName	缺省“录入时间”字段名	缺省值：字段名后缀为“_INPUTTIME”
DefaultDepartmentColumnName	缺省“录入人部门”字段名	缺省值：字段名后缀为“_DEPARTMENT”
DefaultPositionColumnName	缺省“录入人岗位”字段名	缺省值：字段名后缀为“_POSITION”
DefaultInputerAddressColumnName	缺省“录入人地址”字段名	缺省值：字段名后缀为“_INPUTERIP”

当表字段是以上述格式命名的，则这个字段必定是 IsInputerInfoColumn，并强制纳入到 Phenix丶的“自动填充录入信息”服务的管理范围内。

只要业务类中映射了符合上述命名规则的字段，则当新增、修改业务对象的时候，都会自动将当前机器时间、当前用户的工号等信息填充到这些字段中，并持久化（当前用户信息来源于 Phenix.Business.Security.UserPrincipal.User，它代表了当前所登录的用户，在跨域调用服务的时候会作为上下文内容之一(用户验证信息)传递出去；User.Identity 属性包含了当前用户的一些特性，其中就有诸如“UserNumber”等信息）。

由于上述这些字段是由 Phenix丶全程控制的，所以其关联的属性是不允许写的：

```
        /// <summary>
        /// 录入人
        /// </summary>
        public static readonly Phenix.Business.PropertyInfo<string> InputerProperty =
RegisterProperty<string>(c => c.Inputer);
        [Phenix.Core.Mapping.Field(FriendlyName = "录入人", Alias = "DPT_INPUTER", TableName =
"WPL_DELIVERY_PLAN_TYPE", ColumnName = "DPT_INPUTER", NeedUpdate = true, OverwritingOnUpdate = true,
IsInputerColumn = true)]
        private string _inputer;
        /// <summary>
        /// 录入人
        /// </summary>
        [System.ComponentModel.DisplayName("录入人")]
        public string Inputer
        {
            get { return GetProperty(InputerProperty, _inputer); }
        }

        /// <summary>
        /// 录入时间
        /// </summary>
        public static readonly Phenix.Business.PropertyInfo<DateTime?> InputtimeProperty =
RegisterProperty<DateTime?>(c => c.Inputtime);
        [Phenix.Core.Mapping.Field(FriendlyName = "录入时间", Alias = "DPT_INPUTTIME", TableName =
"WPL_DELIVERY_PLAN_TYPE", ColumnName = "DPT_INPUTTIME", NeedUpdate = true, OverwritingOnUpdate = true,
IsInputTimeColumn = true)]
        private DateTime? _inputtime;
        /// <summary>
        /// 录入时间
        /// </summary>
        [System.ComponentModel.DisplayName("录入时间")]
        public DateTime? Inputtime
        {
```

```
        get { return GetProperty(InputtimeProperty, _inputtime); }
    }
```

当然，如果字段命名不符合规范，这也问题不大，就是要多做一步工作，在相应的业务类字段 Phenix.Core.Mapping.FieldAttribute 标签上自行设置对应的标记属性值为 true 即可：

属性	说明	备注
IsInputerColumn	指示该字段是输入人	缺省为 false；当 ColumnName 包含 Phenix.Core.Mapping.CodingStandards.DefaultInputerColumnName 时必定是输入人；
IsDepartmentColumn	指示该字段是部门	缺省为 false；当 ColumnName 包含 Phenix.Core.Mapping.CodingStandards.DefaultDepartmentColumnName 时必定是部门；
IsPositionColumn	指示该字段是岗位	缺省为 false；当 ColumnName 包含 Phenix.Core.Mapping.CodingStandards.DefaultPositionColumnName 时必定是岗位；
IsInputTimeColumn	指示该字段是输入时间	缺省为 false；当 ColumnName 包含 Phenix.Core.Mapping.CodingStandards.DefaultInputTimeColumnName 时必定是入时间；
IsInputerAddressColumn	指示该字段是输入人地址	缺省为 false；当 ColumnName 包含 Phenix.Core.Mapping.CodingStandards.DefaultInputerAddressColumnName 时必定是输入人地址；

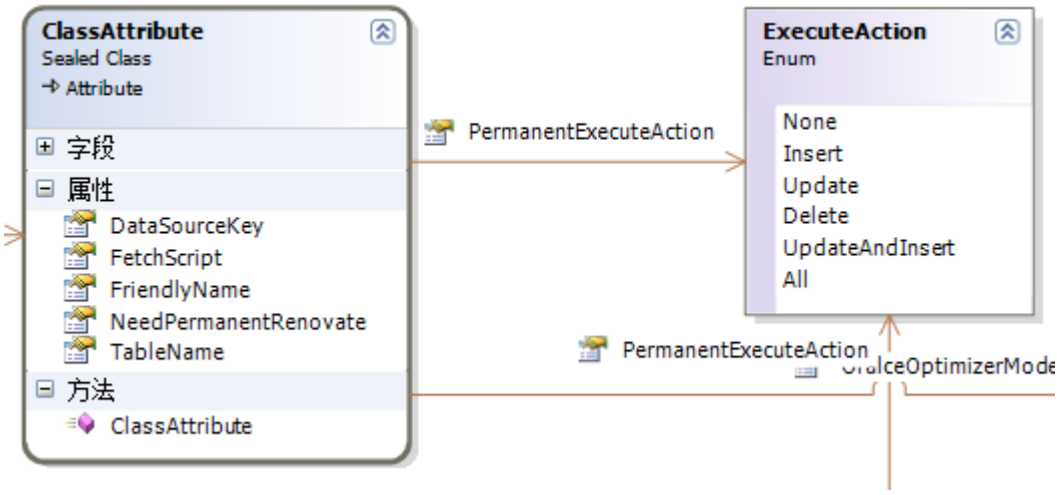
13.1.2自动记录业务数据的增删改痕迹

上述方法是静态地记录了业务数据被操作的行为，那动态的操作痕迹如何记录下来？这需要在业务类上打上标记，告知 Phenix丶在持久化这个业务对象的时候如何记录操作痕迹。

13.1.2.1 配置方法

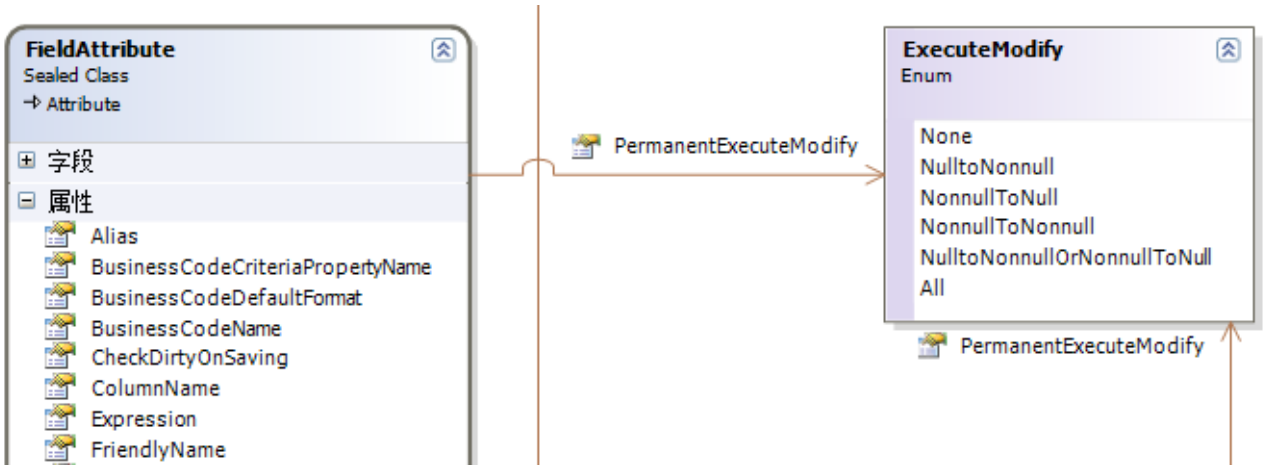
首先在 ClassAttribute 标签上设置 PermanentExecuteAction 属性值，类型为 ExecuteAction，申明当业务数据处于什么持久化状态下才需要记录下痕迹。

ExecuteAction 枚举内容如下图（望文生义，不必解释）：



当 ExecuteAction 申明中含有 ExecuteAction.Update 时，应该在希望留痕的字段 FieldAttribute 标签上设置 PermanentExecuteModify 属性值（类型为 ExecuteModify），申明当业务对象的这些字段值发生改变，且处于哪种新旧值变换状况下才需要记录下痕迹。

ExecuteModify 枚举内容如下图（望文生义，不必解释）：



举例如下：

```
/// <summary>
/// 用户
/// </summary>
[Phenix.Core.Mapping.ClassAttribute("PH_User", FriendlyName = "用户", PermanentExecuteAction = ExecuteAction.All)]
    public abstract class User<T> : Phenix.Business.BusinessBase<T> where T : User<T>

    /// <summary>
    /// 姓名
    /// </summary>
```

```
    public static readonly Phenix.Business.PropertyInfo<string> NameProperty =
RegisterProperty<string>(c => c.Name);
    [Phenix.Core.Mapping.Field(FriendlyName = "姓名", Alias = "US_Name", TableName = "PH_User",
ColumnName = "US_Name", NeedUpdate = true, IsNameColumn = true, InLookUpColumn = true,
InLookUpColumnDisplay = true, PermanentExecuteModify = ExecuteModify.All)]
    private string _name;
    /// <summary>
    /// 姓名
    /// </summary>
    [System.ComponentModel.DisplayName("姓名")]
    public string Name
    {
      get { return GetProperty(NameProperty, _name); }
      set { SetProperty(NameProperty, ref _name, value); }
    }
```

13.1.2.2 持久化的方法

业务数据的操作痕迹，在对象 Save()时会被 Phenix 丶自动记录在表 PH_ExecuteActionLog 里：

```
EA_ID NUMERIC(15) NOT NULL,
EA_Time DATE NOT NULL,                    --时间
EA_UserNumber VARCHAR(10) NOT NULL,       --登录工号
EA_BusinessName VARCHAR(255) NOT NULL,    --业务类名
EA_BusinessPrimaryKey VARCHAR(4000) NULL, --业务主键值
EA_Action NUMERIC(5) NOT NULL,            --执行动作(Phenix.Core.Mapping.ExecuteAction)
EA_Log LONG /*TEXT*/ NULL,                --日志
```

13.1.2.3 检索与删除的方法

我们可以直接操作表来检索它们，也可以通过业务类上的相关方法：

13.1.2.3.1 Phenix.Business.BusinessBase<T>提供的方法

```
    /// <summary>
    /// 检索执行动作
    /// </summary>
    /// <returns>执行动作信息队列</returns>
    public IList<ExecuteActionInfo> FetchExecuteAction()

    /// <summary>
    /// 检索执行动作
    /// userNumber = null
    /// </summary>
    /// <param name="action">执行动作</param>
    /// <param name="startTime">起始时间</param>
    /// <param name="finishTime">结束时间</param>
```

```
    /// <returns>执行动作信息队列</returns>
    public static IList<ExecuteActionInfo> FetchExecuteAction(ExecuteAction action, DateTime startTime, DateTime finishTime)

    /// <summary>
    /// 检索执行动作
    /// </summary>
    /// <param name="userNumber">登录工号，null代表全部</param>
    /// <param name="action">执行动作</param>
    /// <param name="startTime">起始时间</param>
    /// <param name="finishTime">结束时间</param>
    /// <returns>执行动作信息队列</returns>
    public static IList<ExecuteActionInfo> FetchExecuteAction(string userNumber, ExecuteAction action, DateTime startTime, DateTime finishTime)

    /// <summary>
    /// 清除执行动作
    /// userNumber = null
    /// </summary>
    /// <param name="action">执行动作</param>
    /// <param name="startTime">起始时间</param>
    /// <param name="finishTime">结束时间</param>
    public static void ClearExecuteAction(ExecuteAction action, DateTime startTime, DateTime finishTime)

    /// <summary>
    /// 清除执行动作
    /// </summary>
    /// <param name="userNumber">登录工号，null代表全部</param>
    /// <param name="action">执行动作</param>
    /// <param name="startTime">起始时间</param>
    /// <param name="finishTime">结束时间</param>
    public static void ClearExecuteAction(string userNumber, ExecuteAction action, DateTime startTime, DateTime finishTime)
```

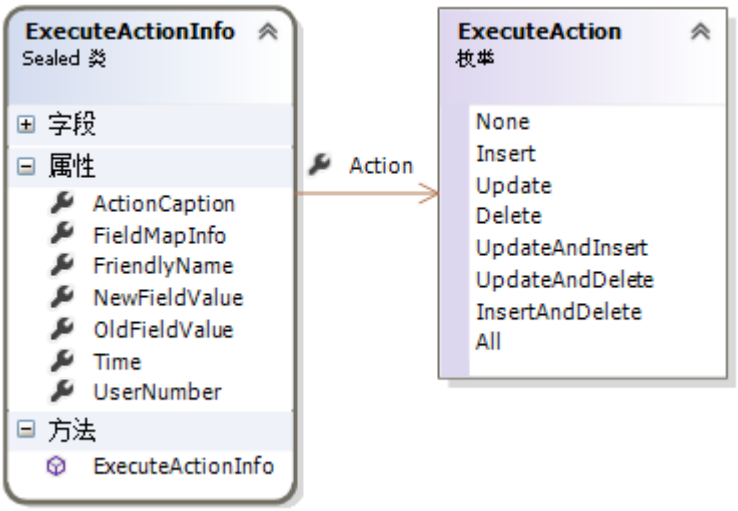
13.1.2.3.2 Phenix.Business.CommandBase<T>提供的方法

```
    /// <summary>
    /// 检索执行动作
    /// </summary>
    /// <returns>执行动作信息队列</returns>
    public IList<ExecuteActionInfo> FetchExecuteAction()

    /// <summary>
    /// 检索执行动作
    /// </summary>
    /// <param name="startTime">起始时间</param>
```

```
    /// <param name="finishTime">结束时间</param>
    /// <returns>执行动作信息队列</returns>
    public static IList<ExecuteActionInfo> FetchExecuteAction(DateTime startTime, DateTime finishTime)

    /// <summary>
    /// 检索执行动作
    /// </summary>
    /// <param name="userNumber">登录工号，null代表全部</param>
    /// <param name="startTime">起始时间</param>
    /// <param name="finishTime">结束时间</param>
    /// <returns>执行动作信息队列</returns>
    public static IList<ExecuteActionInfo> FetchExecuteAction(string userNumber, DateTime startTime,
DateTime finishTime)

    /// <summary>
    /// 清除执行动作
    /// </summary>
    /// <param name="action">执行动作</param>
    /// <param name="startTime">起始时间</param>
    /// <param name="finishTime">结束时间</param>
    public static void ClearExecuteAction(ExecuteAction action, DateTime startTime, DateTime finishTime)

    /// <summary>
    /// 清除执行动作
    /// </summary>
    /// <param name="userNumber">登录工号，null代表全部</param>
    /// <param name="action">执行动作</param>
    /// <param name="startTime">起始时间</param>
    /// <param name="finishTime">结束时间</param>
    public static void ClearExecuteAction(string userNumber, ExecuteAction action, DateTime startTime,
DateTime finishTime)
```

13.1.2.3.3 执行动作信息类 Phenix.Core.Log.ExecuteActionInfo

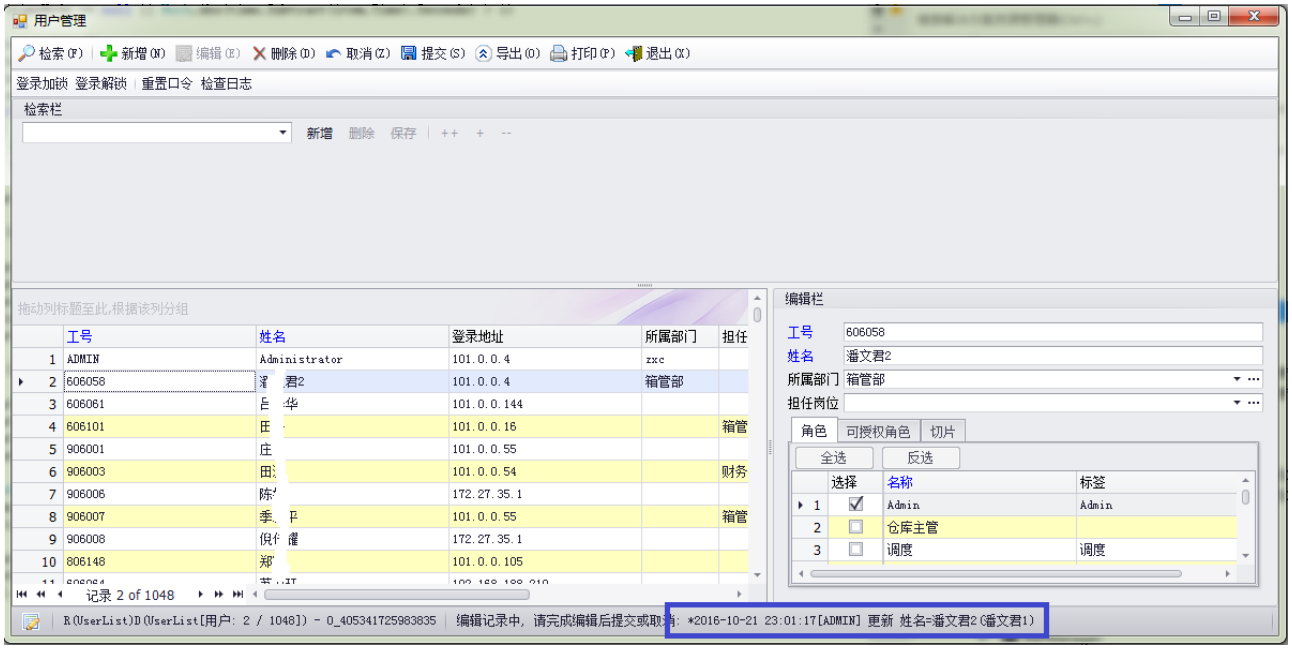
以上函数都用到了 ExecuteActionInfo 对象来返回业务对象的操作痕迹信息，结构如下：



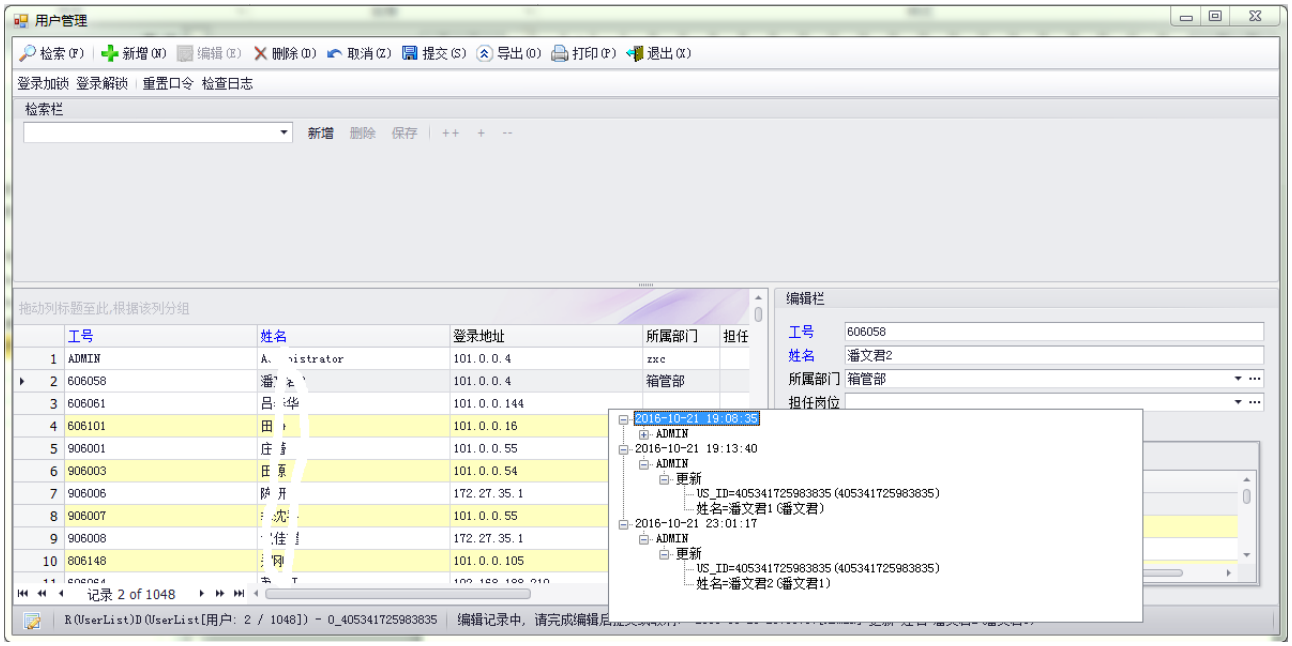
13.1.2.3.4 案例

在 BarManager 组件上已集成了查看日志的功能。

当浏览记录时，状态栏中会自动显示出当前对象的最近一次操作痕迹：



点击它，会弹出窗体供浏览全部的历史痕迹：



核心代码见：

```
namespace Phenix.Windows
{
  partial class ShowExecuteActionForm : Form
  {
    private ShowExecuteActionForm()
    {
      InitializeComponent();
    }

    private static ShowExecuteActionForm _self;

    /// <summary>
    /// 执行
    /// </summary>
    public static void Execute(IBusinessObject business)
    {
      if (_self == null)
        _self = new ShowExecuteActionForm();
      _self.Left = MousePosition.X - _self.Width / 2;
      _self.Top = MousePosition.Y - _self.Height;

      if (_self.Visible)
      {
        _self.Hide();
        return;
      }
```

```
_self.treeView.Nodes.Clear();
if (business != null)
{
  IList<ExecuteActionInfo> infos = business.FetchExecuteAction();
  if (infos != null && infos.Count > 0)
  {
    TreeNode timeNode = null;
    DateTime time = DateTime.Now;
    TreeNode userNumberNode = null;
    string userNumber = null;
    TreeNode actionNode = null;
    ExecuteAction action = ExecuteAction.None;
    foreach (ExecuteActionInfo item in infos)
    {
      if (timeNode == null || Math.Abs(time.Subtract(item.Time).Seconds) > 1)
      {
        time = item.Time;
        timeNode = new TreeNode();
        timeNode.Text = item.Time.ToString();
        _self.treeView.Nodes.Add(timeNode);
        if (userNumberNode != null)
        {
          userNumberNode.ExpandAll();
          userNumberNode = null;
        }
        actionNode = null;
      }
      if (userNumberNode == null || String.CompareOrdinal(userNumber, item.UserNumber) != 0)
      {
        userNumber = item.UserNumber;
        userNumberNode = new TreeNode();
        userNumberNode.Text = item.UserNumber;
        timeNode.Nodes.Add(userNumberNode);
        actionNode = null;
      }
      if (actionNode == null || action != item.Action)
      {
        action = item.Action;
        actionNode = new TreeNode();
        actionNode.Text = item.ActionCaption;
        userNumberNode.Nodes.Add(actionNode);
      }
      TreeNode changeInfo = new TreeNode();
      changeInfo.Text = item.ChangeInfo;
      actionNode.Nodes.Add(changeInfo);
```

```
            }
            if (userNumberNode != null)
              userNumberNode.ExpandAll();
            _self.Show();
          }
        }
      }
    }
}
```

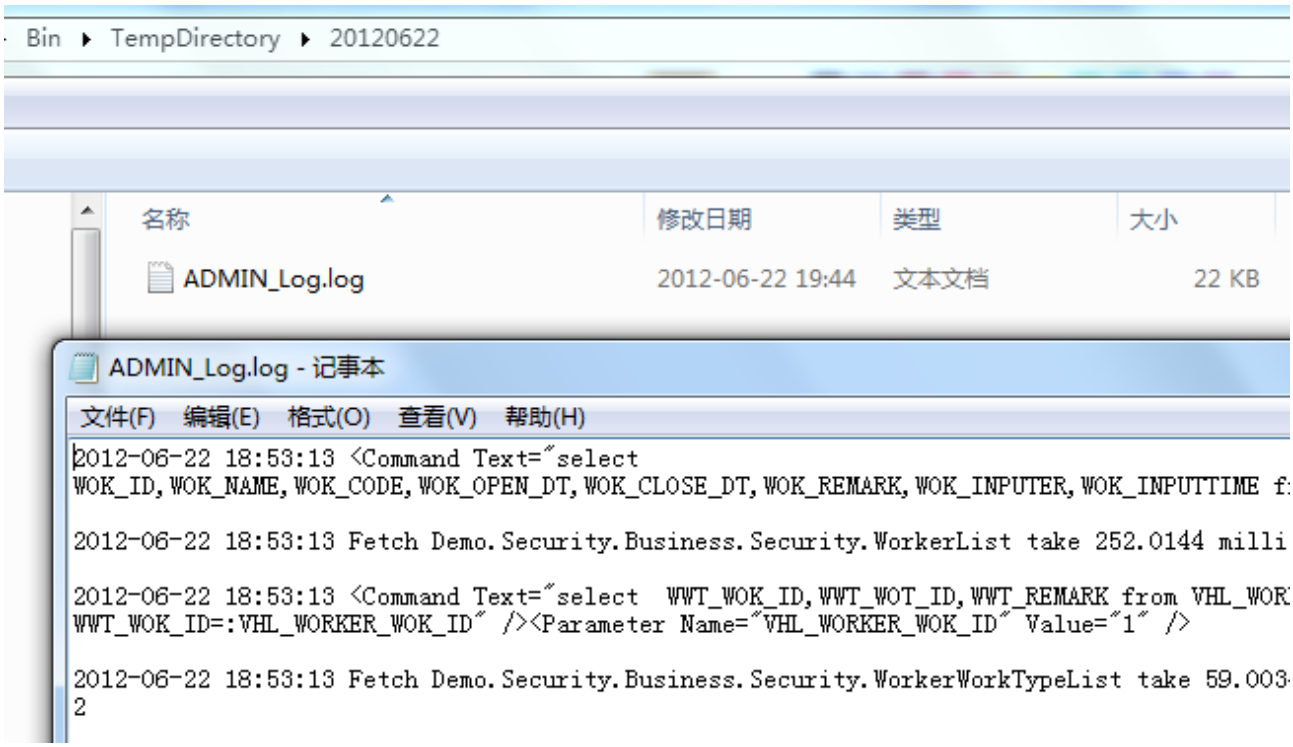
13.2 事件日志

Phenix丶提供了自有的事件日志 EventLog 类：

EventLog 有一个总开关属性：Enabled，缺省为 true，当设置成 false 时，调用 Save()函数将不会有任何动作。

13.2.1辅助开发的日志

要将日志保存在本地（目录 TempDirectory）日志文件中：



可调用 EventLog 类中的如下函数，想存什么就存什么：

```
/// <summary>
/// 保存日志
/// </summary>
/// <param name="log">日志</param>
public static void Save(string log)

/// <summary>
/// 保存日志
/// </summary>
/// <param name="log">日志</param>
/// <param name="extension">后缀</param>
public static void Save(string log, string extension)

/// <summary>
/// 保存错误日志
/// </summary>
/// <param name="log">日志</param>
/// <param name="error">错误</param>
public static void Save(string log, Exception error)

/// <summary>
/// 保存错误日志
/// </summary>
/// <param name="log">日志</param>
```

```
/// <param name="error">错误</param>
/// <param name="extension">后缀</param>
public static void Save(string log, Exception error, string extension)

/// <summary>
/// 保存错误日志
/// </summary>
/// <param name="sender">发送者</param>
/// <param name="error">错误</param>
public static void Save(object sender, Exception error)

/// <summary>
/// 保存错误日志
/// </summary>
/// <param name="sender">发送者</param>
/// <param name="error">错误</param>
/// <param name="extension">后缀</param>
public static void Save(object sender, Exception error, string extension)

/// <summary>
/// 保存错误日志
/// </summary>
/// <param name="method">函数的信息</param>
/// <param name="log">日志</param>
/// <param name="error">错误</param>
public static void Save(MethodBase method, string log, Exception error)

/// <summary>
/// 保存错误日志
/// </summary>
/// <param name="method">函数的信息</param>
/// <param name="log">日志</param>
/// <param name="error">错误</param>
/// <param name="extension">后缀</param>
public static void Save(MethodBase method, string log, Exception error, string extension)
```

13.2.2辅助维护的日志

应用系统上线后，维护日志必须存放在数据库中，这样才能方便检索。否则，如按上述方法，日志将会散落在每台机器上，是无法收集、整理和分析的。

存储日志的表 PH_ExecuteLog 结构如下：

```
EL_ID NUMERIC(15) NOT NULL,
EL_Time DATE NOT NULL,                    --时间
EL_UserNumber VARCHAR(10) NOT NULL,       --登录工号
```

```
EL_BusinessName VARCHAR(255) NOT NULL,   --业务类名
EL_Message VARCHAR(4000) NULL,           --消息
EL_ExceptionName VARCHAR(255) NULL,      --错误名
EL_ExceptionMessage VARCHAR(4000) NULL,  --错误消息
```

请注意：当系统处于测试状态（Phenix.Core.AppConfig.Debugging = true），或者当前用户带 ADMIN 权限（user.IsAdminRole = true），则仅保存到本地日志文件中。

13.2.2.1 保存日志

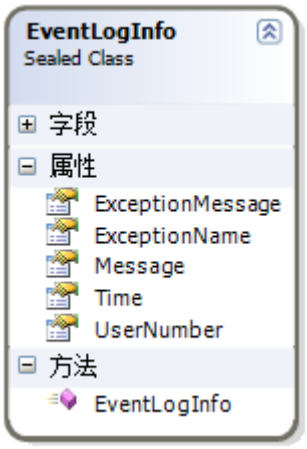
```
/// <summary>
/// 保存对象消息
/// </summary>
/// <param name="user">登录用户</param>
/// <param name="objectType">类</param>
/// <param name="log">日志</param>
public static void Save(IPrincipal user, Type objectType, string log)

/// <summary>
/// 保存对象消息
/// </summary>
/// <param name="user">登录用户</param>
/// <param name="objectType">类</param>
/// <param name="error">错误</param>
public static void Save(IPrincipal user, Type objectType, Exception error)

/// <summary>
/// 保存对象消息
/// </summary>
/// <param name="user">登录用户</param>
/// <param name="method">函数的信息</param>
/// <param name="error">错误</param>
public static void Save(IPrincipal user, MethodBase method, Exception error)
```

13.2.2.2 检索日志

```
/// <summary>
/// 检索日志消息
/// </summary>
/// <param name="userNumber">登录工号，null代表全部</param>
/// <param name="objectType">类，null代表全部</param>
/// <param name="startTime">起始时间</param>
/// <param name="finishTime">结束时间</param>
/// <returns>日志队列</returns>
public static IList<EventLogInfo> Fetch(string userNumber, Type objectType, DateTime startTime,
```

```
DateTime finishTime)
```

上述函数中，日志消息是通过填充到 EventLogInfo 对象来返回的：



13.2.2.3 删除日志

```
/// <summary>
/// 清除日志消息
/// </summary>
/// <param name="userNumber">登录工号，null代表全部</param>
/// <param name="objectType">类，null代表全部</param>
/// <param name="startTime">起始时间</param>
/// <param name="finishTime">结束时间</param>
public static void Clear(string userNumber, Type objectType, DateTime startTime, DateTime finishTime)
```