20 动态刷新

动态刷新技术所要解决的问题，是在尽可能短的时间（3 秒）内能检索到实时变化的业务数据。我们不可能在间隔很短的时间（每隔 1、2 秒）内重复 Fetch 业务数据，尤其是大批量的业务数据，因为这会给数据库带来很大的压力，也受到了网络环境的限制，特别是搭建在广域、无线网络上的应用系统。

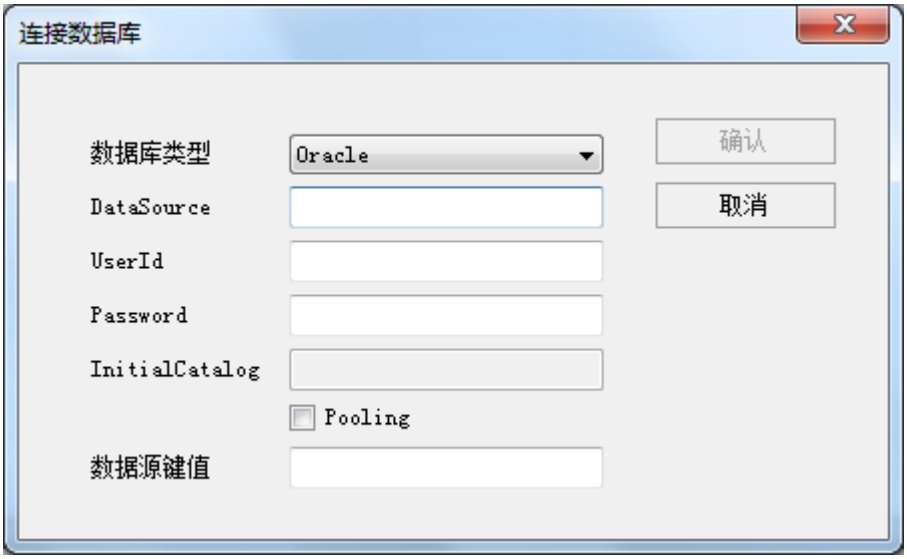
动态刷新技术由动态刷新服务程序（Phenix.Renovate.Server.exe）和客户端核心程序集（Phenix.Renovate.Client.Core.dll）以及客户端组件包（Phenix.Renovate.Client.dll）组成，服务程序维持着最新版本的业务数据集合，并实时响应客户端的动态刷新数据请求，利用数据推送技术，使得客户端始终能保持最新版本的业务数据。

利用动态刷新技术这种特性，我们可以将那些供浏览、供选择的只读业务集合对象转换为动态刷新业务集合对象，整体性能会比直接 Fetch 要高。

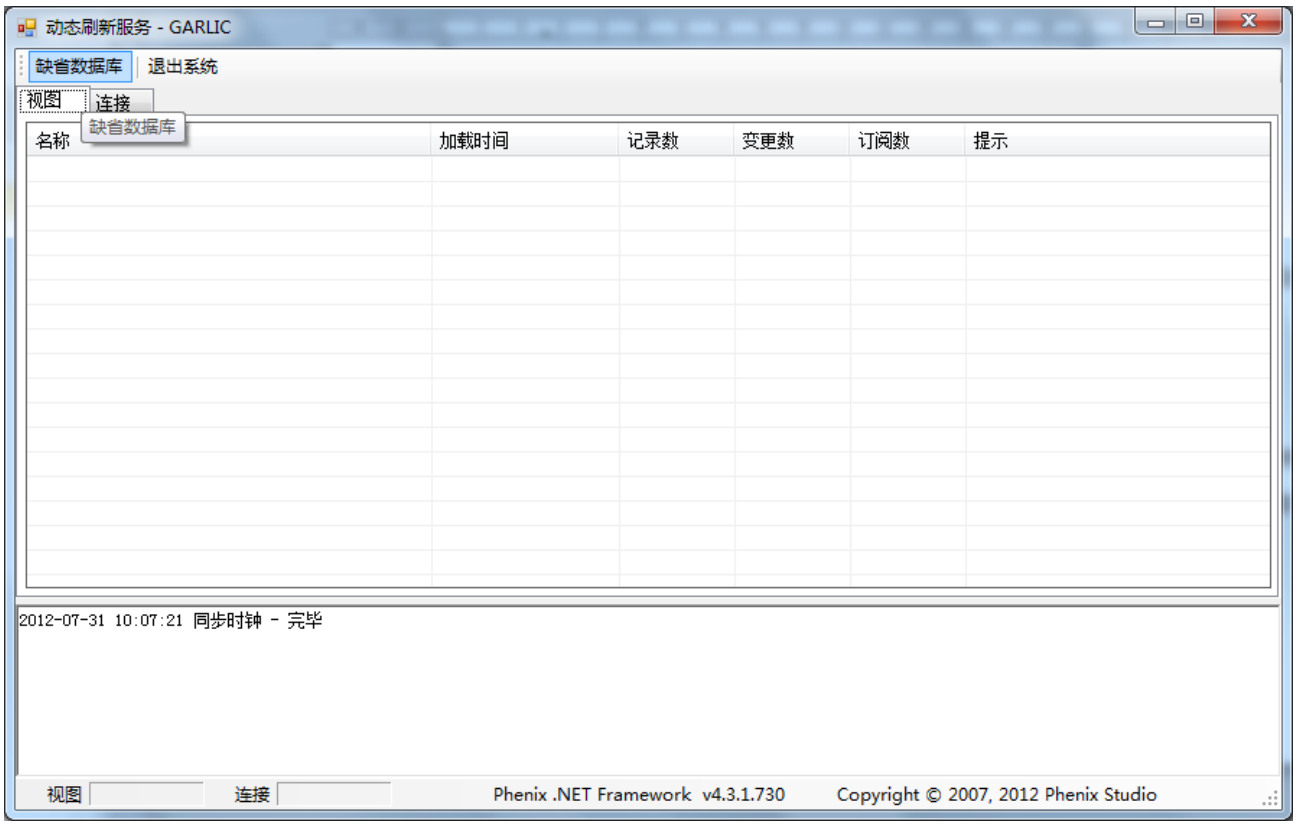
20.1 动态刷新服务程序

20.1.1连接数据库

动态刷新服务程序(Phenix.Renovate.Server.exe)在第一次启动时，会弹出连接数据库的对话框：



只要数据库连接成功，下次启动时就会维持当前的连接参数自动连接上数据库。如果需要修改连接参数，可通过“缺省数据库”菜单来点出上述对话框重新配置：

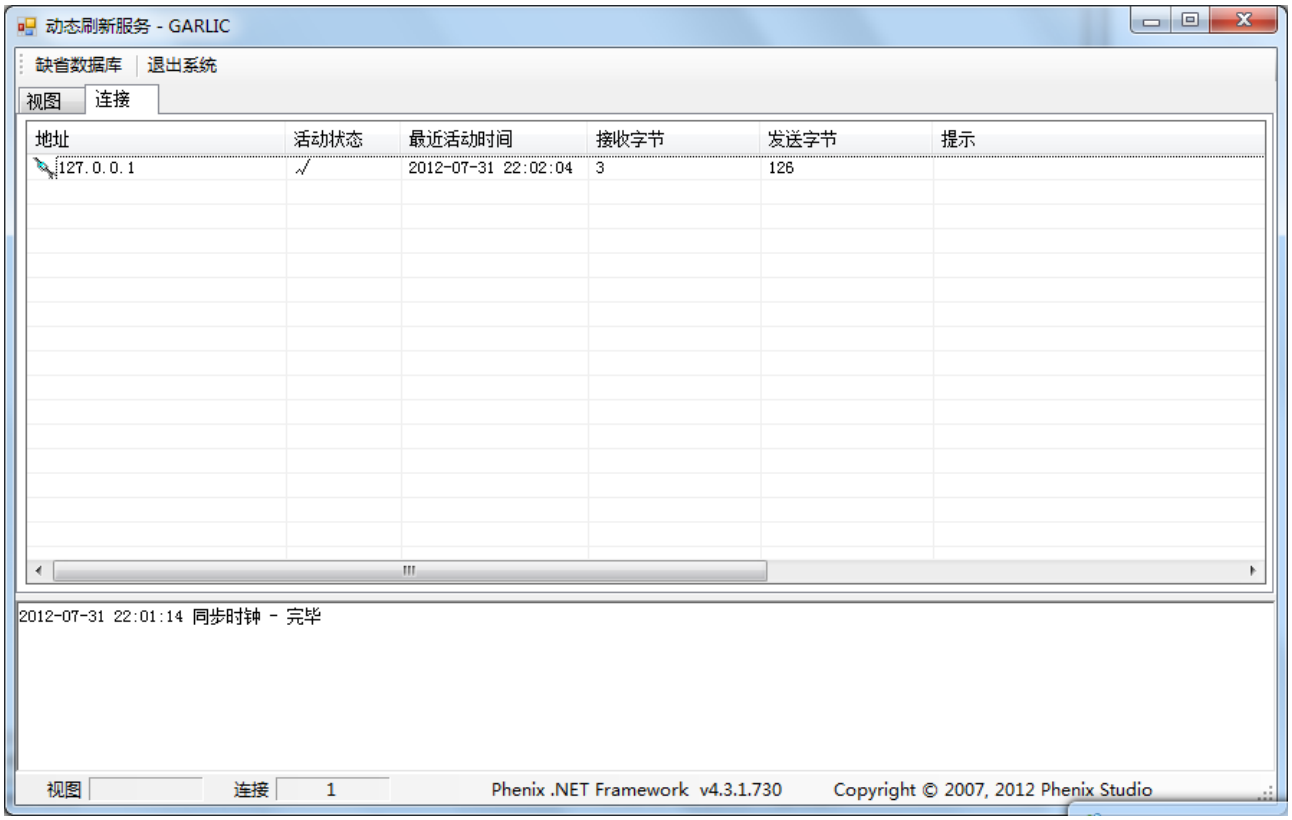


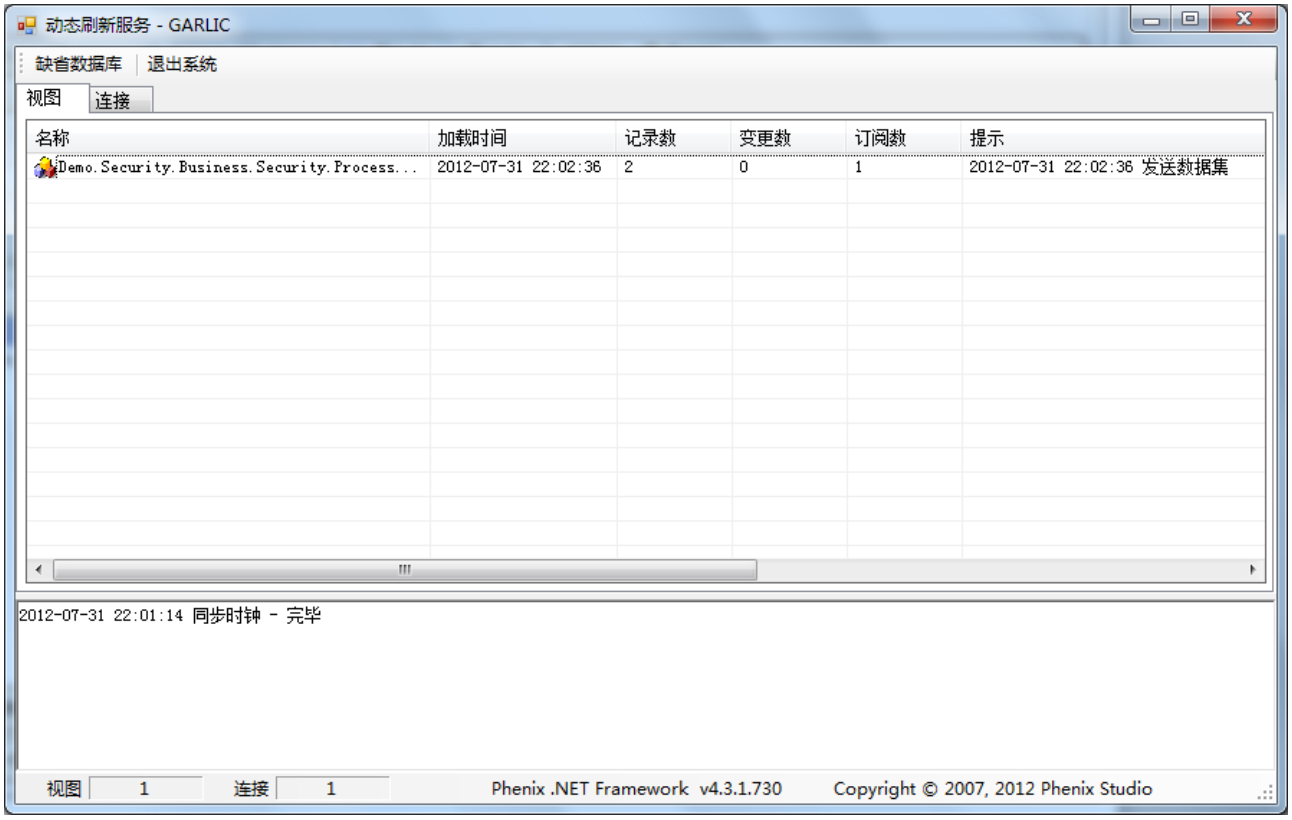
注意：目前动态刷新技术仅适用于 Oracle 数据库，后续版本会针对 SqlServer 做升级。

20.1.2监控界面

服务程序启动后，即进入自动运行状态，无需人工干预。

服务程序仅提供了动态刷新的监控界面：在“视图”、“连接”两页中，实时显示当前有哪些业务数据集被维持着最新版本，有哪些需提供动态刷新服务的客户端的 IP 地址、收/发的数据流量数：

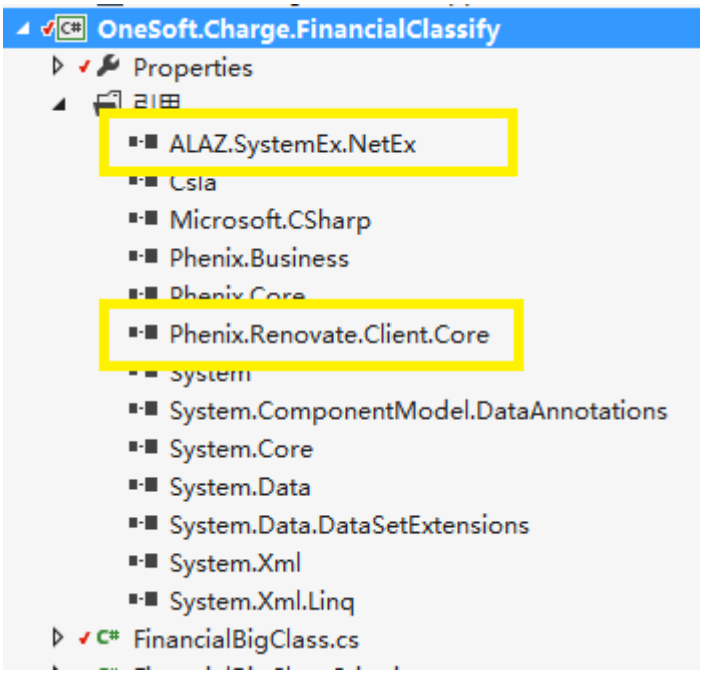

动态刷新服务 - GARLIC
缺省数据库
退出系统
视图
连接
地址
活动状态
最近活动时间
接收字节
发送字节
提示
127.0.0.1
2012-07-31 22:02:04
3
126
2012-07-31 22:01:14 同步时钟 - 完毕
视图
连接
1
Phenix .NET Framework v4.3.1.730
Copyright © 2007, 2012 Phenix Studio


动态刷新服务 - GARLIC
缺省数据库
退出系统
视图
连接
名称
加载时间
记录数
变更数
订阅数
提示
Demo.Security.Business.Security.Process...
2012-07-31 22:02:36
2
0
1
2012-07-31 22:02:36 发送数据集
2012-07-31 22:01:14 同步时钟 - 完毕
视图
1
连接
1
Phenix .NET Framework v4.3.1.730
Copyright © 2007, 2012 Phenix Studio

20.2 动态刷新客户端组件

20.2.1开启、关闭、暂停动态刷新

应用系统如果要使用动态刷新技术的话，请在客户端主窗体的工程中引用如下程序集：



然后在主窗体中添加以下两事件，并执行黄底黑字部分的代码：

```
private void MainForm_Shown(object sender, EventArgs e)
{
  //启动动态刷新
  Phenix.Renovate.Client.Core.Subscriber.Default.Start();
}

private void MainForm_FormClosed(object sender, FormClosedEventArgs e)
{
  //关闭动态刷新
  Phenix.Renovate.Client.Core.Subscriber.Default.Close();
}
```

如需在程序运行当中暂停动态刷新功能，可执行以下代码：

```
Phenix.Renovate.Client.Core.Subscriber.Default.Suspend();
```

20.2.2注册动态刷新服务

业务集合基类 Phenix.Business.BusinessListBase<T, TBusiness>已实现了动态刷新数据的解析功能（Phenix.Core.Renovate.IAnalyser 接口），所以我们只要调用 Phenix.Renovate.Client.Core 程序集 Subscriber.Default 对象提供的 Register()函数，接着再调用它的 Fetch()函数,就可以获取到一个维持最新版本业务数据的动态刷新业务集合对象：

```
/// <summary>
/// 注册
/// </summary>
public void Register<T>()
  where T : IAnalyser

/// <summary>
/// 注册
/// </summary>
/// <param name="synchronizeSpaceSecond">同步间隔(秒)，大于0才执行同步</param>
public void Register<T>(int synchronizeSpaceSecond)
  where T : IAnalyser

/// <summary>
/// 注册
/// </summary>
/// <param name="analyserType">类型</param>
/// <param name="synchronizeSpaceSecond">同步间隔(秒)，大于0才执行同步</param>
public void Register(Type analyserType, int synchronizeSpaceSecond)
```

同步间隔“synchronizeSpaceSecond”参数，单位是秒，当大于 0 秒时才执行同步，缺省为 0 秒。同步间隔，指的是要求动态刷新服务如何定时与数据库数据进行同步，这适用于分组等统计数据的刷新，因为它无法通过逐条记录来更新数据。

Phenix.Renovate.Client.Core 程序集的 Subscriber.Default 缓存了这一个动态刷新业务集合对象，应用系统可以多次调用 Register()函数，返回的将是这同一个对象。通过这种方式，我们可以在进程的多处业务场景中使用到同一个动态刷新业务集合对象。

获取动态刷新业务集合对象的方法：

```
/// <summary>
/// 构建
/// </summary>
public T Fetch<T>()
  where T : IAnalyser
```

```
/// <summary>
/// 构建
/// </summary>
    public IAnalyser Fetch(Type analyserType)
```

20.2.3注销动态刷新服务

如果在进程的一个业务场景中不再使用动态刷新业务集合对象的话（比如关闭了窗体），则应该调用 Phenix.Renovate.Client.Subscriber 的 Unregister()函数进行注销：

```
/// <summary>
/// 注销
/// </summary>
public void Unregister<T>()
  where T : IAnalyser
```

注销动态刷新服务，并不代表动态刷新业务集合对象停止了刷新数据，只是通过调用 Unregister()函数告知动态刷新服务减少了一个订阅者。只有当调用 Unregister()函数的次数与调用 Register()函数的次数相等，Phenix.Renovate.Client.Subscriber 才会停止刷新数据、彻底丢弃这个动态刷新业务集合对象。因此，调用 Register()函数和调用 Unregister()函数应该一对一匹配。

20.2.4业务集合类可供拦截的动态刷新事件函数

业务集合类提供有如下的事件函数，用于拦截动态刷新事件：

```
/// <summary>
/// 分析数据集资料
/// </summary>
protected virtual bool AnalyseDataInfo(DataTable data)

/// <summary>
/// 分析动态刷新资料
/// </summary>
 protected virtual Phenix.Core.Mapping.IEntity AnalyseRenovateInfo(ExecuteAction action, object[] values)
```

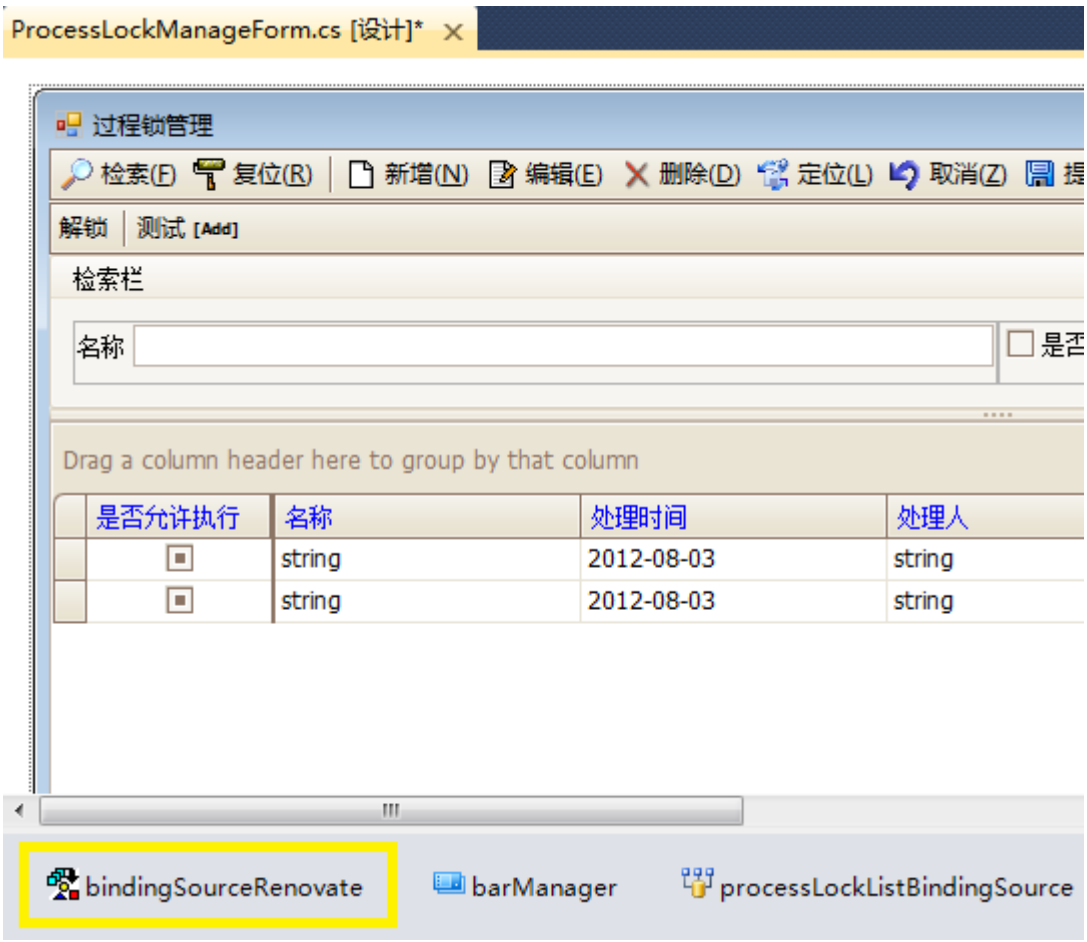
具体使用方法，请见工程：Phenix.Test.使用指南.20。

20.2.5动态刷新辅助组件

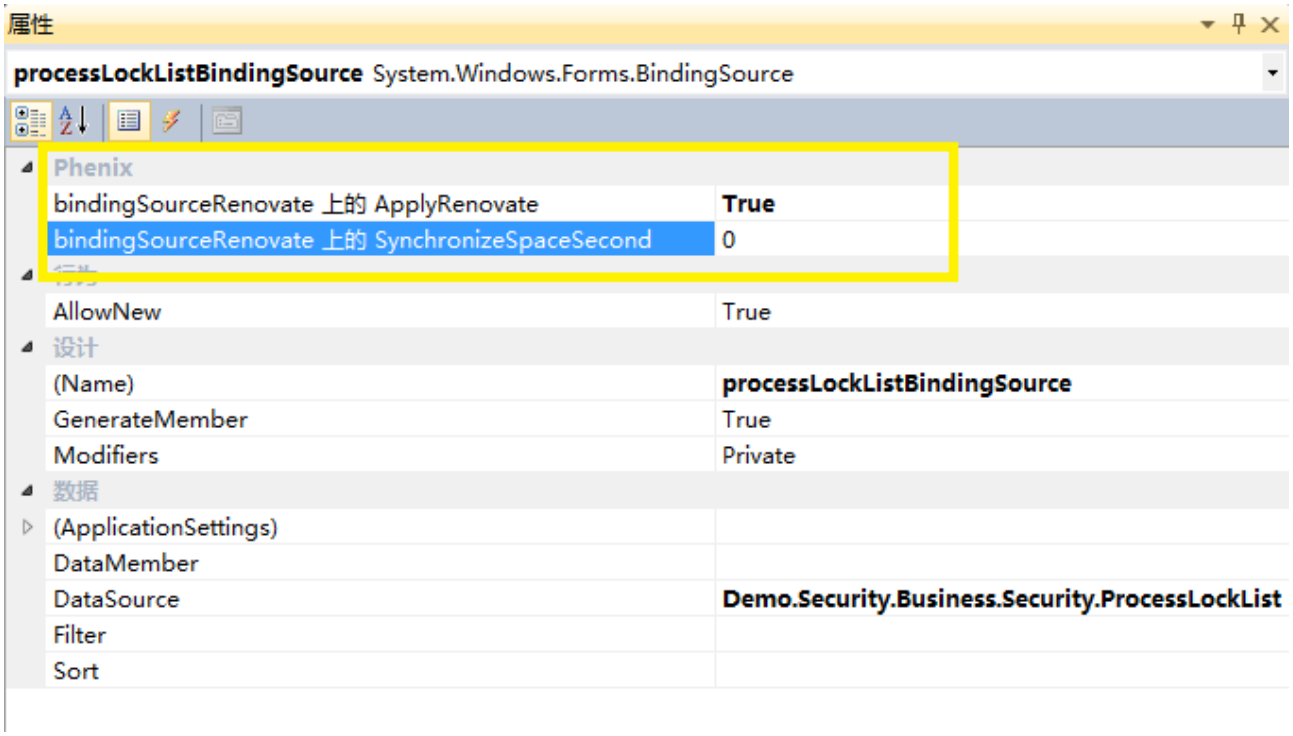
为了提高动态刷新界面设计的开发效率，Phenix丶将上述代码编写过程（注册、注销动态刷新服务）

做了封装，集成到 Phenix.Renovate.Client 程序集的 BindingSourceRenovate 组件中。

首先在 IDE 中将 BindingSourceRenovate 组件拖到需要动态刷新机制的界面设计框中：



所有的 BindingSource 属性栏中都会出现被 BindingSourceRenovate 组件扩展出来的属性：



然后将其扩展属性“ApplyRenovate”值设置为 true 即可实现动态刷新。

在应用系统运行期间，当打开本界面的时候，BindingSourceRenovate 组件会根据其绑定的 DataSource 所申明的业务集合类型自动注册动态刷新服务，并将注册时获得的动态刷新业务集合对象赋值给 DataSource；在关闭本界面的时候，BindingSourceRenovate 组件会自动注销动态刷新服务。

20.3 触发动态刷新的方法

客户端上实时更新的业务数据是由动态刷新服务程序推送过来的，那么，服务程序上的业务数据变更信息又是从何而来的呢？

我们知道，之所以要使用动态刷新技术，实质就是为了实时感知数据库记录的增删改事件。这个事件，在应用系统中的两个层面上会产生：持久层上业务对象的数据提交、数据层上存储过程或直接手工操作表记录的数据提交。下面，我们分别讨论在这两种提交数据的层面上，将如何来触发动态刷新方法。

20.3.1业务对象的数据提交

只要通过 Phenix丶持久层自动提交业务数据的，都适用于本方法。

有一个特例是那些被自动级联删除的表，是不会被动态刷新功能支持。但这不会影响正常的应用，因为在系统设计中，我们往往考虑的是 root 业务对象的动态刷新。

我们仅需在业务类上的 ClassAttribute 标签里设置 NeedPermanentRenovate 属性值等于 true 即可达到目的。

```
/// <summary>
/// 过程锁
/// </summary>
[Phenix.Core.Mapping.ClassAttribute("PH_PROCESSLOCK", NeedPermanentRenovate = true, FriendlyName = "过程锁",), Serializable]
public abstract class ProcessLock<T> : Phenix.Business.BusinessBase<T> where T : ProcessLock<T>
{
  …

}
```

被动态刷新的业务对象，一般选择的是 root，所以标签只要打在这些类上即可。当 Phenix 持久化这个业务对象的时候，会被通知到动态刷新服务系统。

20.3.2当存储过程、触发器中执行了表记录增删改、或通过数据库工具手工修改了表记录

当直接操作数据库完成数据更新后，应该在存储过程、触发器或者数据库工具中，直接运行一次以下清单中的存储过程（传入参数为被修改数据表的表名、被修改记录的 RowId），以通知到 Phenix：

存储过程	参数	说明
PH_Record_Has_Inserted	i_TableName：表名称 i_ROWID：表记录的 ROWID	新增记录后
PH_Record_Has_Updated	i_TableName：表名称 i_ROWID：表记录的 ROWID	修改记录后
PH_Record_Has_Deleted	i_TableName：表名称 i_ROWID：表记录的 ROWID	删除记录后

举例，假设直接在数据库中对 PH_PROCESSLOCK 表新增了 RowId 为“AAAHhBAABAAANayAAA”的记录，那么，我们可以在数据库中执行如下语句，就能通知到动态刷新服务程序：

```
PH_Record_Has_Inserted("PH_PROCESSLOCK", "AAAHhBAABAAANayAAA")
```