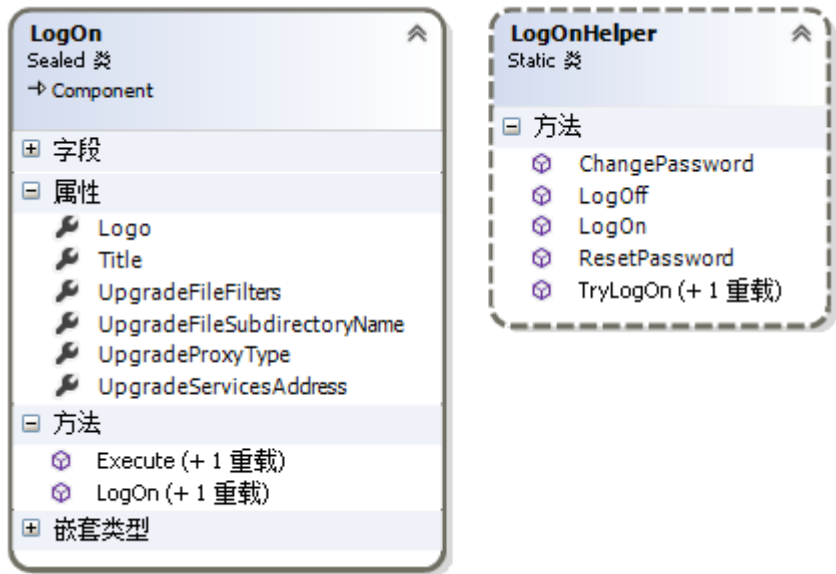


4 登录应用系统

系统登录模块是整个企业级应用系统的用户验证和操作入口，Phenix为此提供了标准的 WinForm 登录组件及其可扩展、自定义的界面开发接口，像 LogOnHelper 更底层的基础接口也适用于 WebForm 等不同展现层的应用开发：



4.1 标准的 WinForm 登录组件

4.1.1 LogOn 组件

所在程序集：Phenix.Services.Client

命名空间：Phenix.Services.Client.Security

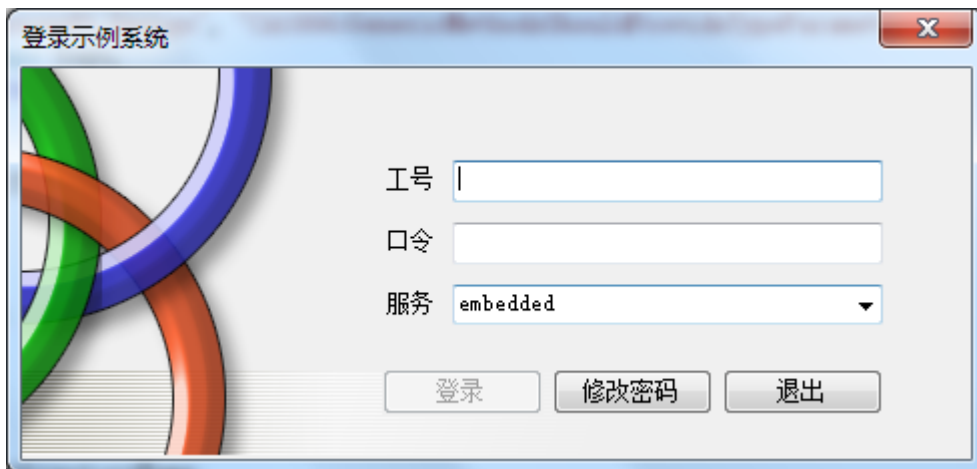
组件功能：提供标准的 WinForm 登录界面，校验用户、修改口令及升级服务。

属性	说明	备注
Title	登陆界面的标题	
Logo	登陆界面的标志	
UpgradeProxyType	升级文件的代理类型	缺省为 null，代表将默认使用 enix.Core.Net.NetConfig.ProxyType;
UpgradeServicesAddress	升级文件的服务地址	缺省为 null，代表将默认使用 Phenix.Core.Net.NetConfig.ServicesAddress;
UpgradeFileSubdirectoryName	升级文件所在服务的子目录名	缺省为 null，代表将默认使用 Phenix.Core.AppConfig.CLIENT_LIBRARY_SUBDIR

4.1.2 LogOn 组件配置及调用方法示例

```
using(LogOn logOn = new LogOn())
{
    logOn.Title = "登录XXX系统"; //title可更换
    logOn.UpgradeFileFilters.Add("*.dll");
    logOn.UpgradeFileFilters.Add("*.wav");
    logOn.UpgradeFileFilters.Add("*.xls");
    Phenix.Core.Security.IPrincipal user = logOn.Execute();
    if (user != null && user.Identity.IsAuthenticated)
    {
        Phenix.Core.Plugin.PluginHost.Default.SendSingletonMessage("Phenix.Windows.Main", null);
    }
}
```

调用 Execute() 函数弹出登录对话框，效果如下：



注：工号输入框的内容，默认下是会保留最近一次登录用户的工号，但也可以通过设置 `logOn.UserNumberCached = false` 来禁用本功能。

4.1.3 可配置化

除了通过编写代码对 `LogOn.Title` 属性的赋值来改变登录界面的标题外，还可以通过改写 `Phenix.Services.Client.dll.config` 文件中的 `<Phenix.Services.Client.Properties.Settings>` 配置内容来改变登录界面上所有控件的标签。

界面背景 logo 也是可以替换的：

```
logOn.Logo = System.Drawing.Image.FromFile("Phenix.logo.jpg");
```

4.1.4 指定自动升级的文件

示例中添加到 logOn.UpgradeFileFilters 队列中的是升级文件的搜索模式，升级文件存放在服务程序（Host）所在目录的 ClientLibrary 子目录里，只要与搜索模式匹配且符合下述条件的文件即被下载并覆盖客户端文件：

- 服务端文件与客户端文件的大小不一致；
- 服务端文件的修改时间晚于客户端文件；

本方法可应用于在一个系统中有多种不同的客户端，但需连接到同一个 Host 服务，要各自有选择性下载客户端文件的场景。比如：

```
using(LogOn logOn = new LogOn())
{
    logOn.UpgradeFileFilters.Add("车载*.dll");

    IPrincipal user = logOn.Execute<LogOnDialog>();
    if (user != null && user.Identity.IsAuthenticated)
    {
        ...
    }
}
```

4.1.5 指定升级服务的地址

LogOn 组件默认从登录的 Host 下载文件升级客户端，但也可以通过指定的 Host 获得专门的升级服务。

以下代码摘自“Phenix.Test.使用指南.04.1.5”工程，可运行试用、观察效果：

```
using(LogOn logOn = new LogOn())
{
    logOn.Title = "登录XXX系统"; //title可更换
    //logOn.Logo = System.Drawing.Image.FromFile("Phenix.logo.jpg"); //logo可更换
    //logOn.UpgradeProxyType = Phenix.Core.Net.NetConfig.ProxyType; //升级文件的代理类型
    logOn.UpgradeServicesAddress = "127.0.0.1"; //升级文件的服务地址
    //logOn.UpgradeFileSubdirectoryName = AppConfig.CLIENT_LIBRARY_SUBDIRECTORY_NAME; //升级文件
    //所在服务的子目录名
    logOn.UpgradeFileFilters.Add("*.dll");
    logOn.UpgradeFileFilters.Add("*.xls");
}
```

```
IPrincipal user = logOn.Execute<LogOnDialog>();
if (user != null && user.Identity.IsAuthenticated)
{
    Console.WriteLine("成功登录到: " + Phenix.Core.Net.NetConfig.ServicesAddress);
    Console.WriteLine("下载服务为: " + logOn.UpgradeServicesAddress);
    Console.WriteLine("请观察是否有文件被下载到" + Phenix.Core.AppConfig.BaseDirectory + "
目录下? ");
    Console.WriteLine();
    break;
}
```

另，以上示例中 LogOnDialog、LogOn 的属性 UpgradeFileSubdirectoryName 已被最新版 Phenix 废弃。如需用到类似功能，请操作 Host 服务程序 SystemInfo 配置界面 System 页上的“客户端下载目录”功能。Host 会为此目录自动添加“Own”子目录，用于针对个别部门群体的个性化下载需求。实施人员须为这些用户配置上所属部门，所属部门的最顶层（一级）部门的代码，将作为“Own”下某子目录的名字（由实施人员添加这个子目录），然后就可以把需要个性化下载的文档拷贝到这子目录里。举例：假设 Host 为客户端提供的下载目录是“\\192.168.1.100\ClientLibrary\”，Host 会自动为其添加“Own”子目录，再假设登录用户所属部门的顶层部门代码为“SGWY”，则其个性化下载的文档应该被部署在“\\192.168.1.100\ClientLibrary\Own\SGWY\”目录。而且，虽有同名文件存放在这不同的目录里（比如“...\ClientLibrary\Test.dll”和“...\ClientLibrary\Own\SGWY\Test.dll”），部门用户的客户端将优先下载自己部门里的文件（比如所属部门代码为“SGWY”的登录用户，其客户端将下载“...\ClientLibrary\Own\SGWY\Test.dll”文件）。

4.2 灵活的个性化设计

LogOn 组件允许你通过继承 Phenix.Services.Client.Security.LogOnDialog 窗体类实现个性化的登录界面：

```
internal partial class LoginDialog : Phenix.Services.Client.Security.LogOnDialog
{
    public LoginDialog()
        : base()
    {
        InitializeComponent();
    }
}
```

界面设计：



调用代码:

```
using (LogOn logOn = new LogOn())
{
    logOn.Title = "登录XXX系统"; //title可更换

    logOn.UpgradeFileFilters.Add("*.dll");
    logOn.UpgradeFileFilters.Add("*.wav");
    logOn.UpgradeFileFilters.Add("*.xls");
    logOn.UpgradeFileFilters.Add("*.xml");
    logOn.UpgradeFileFilters.Add("*.msi");

    IPrincipal user = logOn.Execute<LogOnDialog>();
    if (user != null && user.Identity.IsAuthenticated)
    {
        //启动主Form
        PluginHost.Default.SendSingletonMessage("Phenix.Windows.Main", null);
    }
}
```

具体方法见“Phenix.Windows.Client”工程。

4.3 LogOnHelper 基础接口



更加底层的接口是 Phenix.Services.Client.Security.LogOnHelper 类, 我们可以自行封装它们(比如搭建异构系统), 以最大限度地享用到 Phenix 所提供的服务。

以下是不需要弹出登录界面直接验证用户的代码示例(摘录自 Phenix.StandardPush.Test 工程):

```
while (true)
try
{
    DataSecurityContext context;
    if (LogOnHelper.TryLogOn("127.0.0.1", "ADMIN", password, out context))
        break;
    Console.WriteLine("登录未成功: " + context.Message);
    Console.Write("请重新登录, 输入ADMIN的口令: ");
    password = Console.ReadLine();
}
catch (Exception ex)
{
    Console.WriteLine(Phenix.Core.AppUtilities.GetErrorHint(ex));
    Console.Write("请调整好后点回车继续: ");
    Console.ReadLine();
}
```

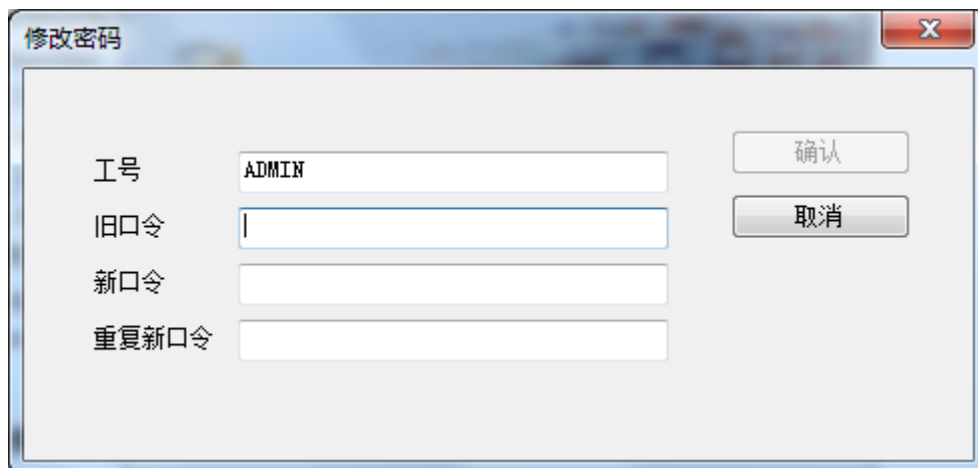
4.4 如何第一次登陆应用系统？



可编译示例工程 Phenix.Windows.Client 后执行它，启动示例系统的登录界面。

Phenix 平台在搭建配置库表（见上文）的时候，会在配置表 Ph_User 中添加一条 ADMIN 用户（初始登录密码与用户名相同，即 ADMIN，注意是大写）记录，所以我们可以用 ADMIN 用户来试用如何登录示例系统。

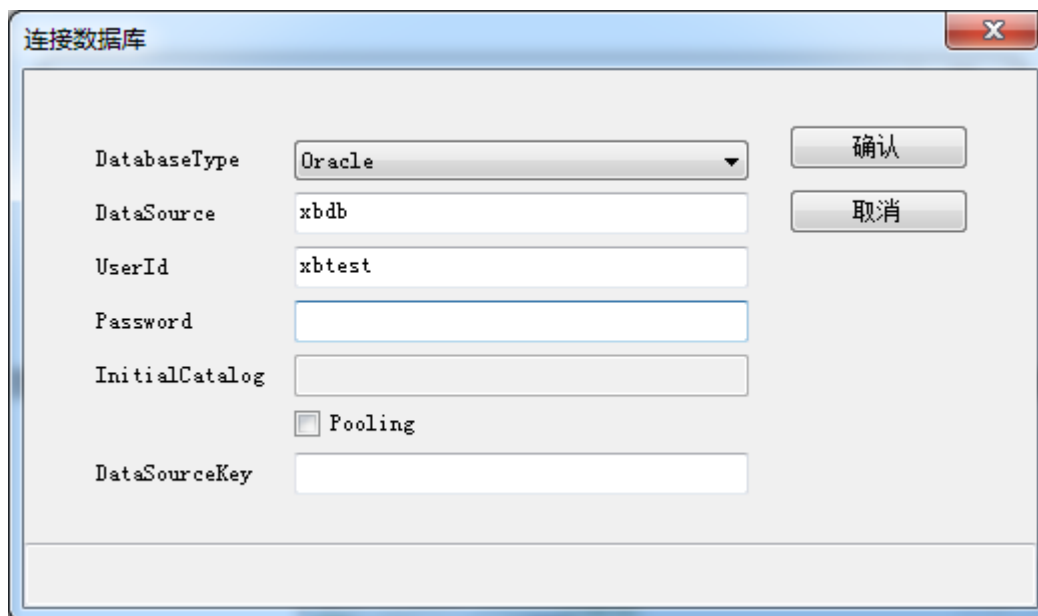
如要修改登录用户的密码，在登录界面上有“修改密码”功能按钮，点击后会弹出如下对话框：



登录界面的“服务”输入框，可以输入服务端 IP 地址，也可以清空内容来自动填充成“embedded”值，即切换为 2 层物理架构（直连数据库）：



鉴于安全考虑，每次直连数据库都需输入连接串信息：



2 层物理架构仅适用于测试阶段，在正式环境下必须部署为 3 层物理架构。

3 层物理架构下，服务器上应事先启动服务端 Phenix.Services.Host.x86/x64.exe 程序，并能保证其开放的端口（默认协议是 Remoting TCP 8086/8087 端口）能穿透从客户端到服务端的每道防火墙。

如果服务端的端口调整为非默认端口号，在客户端的登录界面上，可以在“服务”输入框中填写服务 IP 地址后跟端口号，分隔符为“:”，即输入格式为“IP:Port”。比如：192.168.250.101:8086。