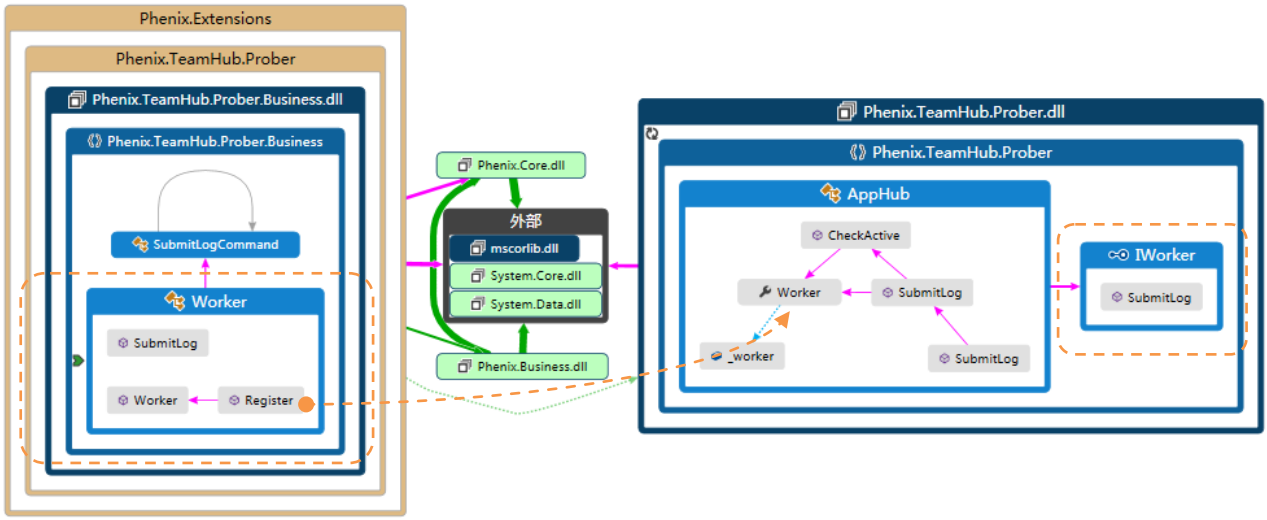
2 缺陷管理

在软件过程里，缺陷管理是很重要的一个环节，它除了可以对需求的完成度进行控制，同时也可以对软件本身的质量进行控制，以保证软件开发迭代的顺利进行。Phenix丶的 Phenix Teamwork Tools 工具包，致力于为缺陷管理提供全系列的自动化、智能化的辅助平台。

2.1 缺陷采集

2.1.1 框架介绍

缺陷管理首要的任务是缺陷的采集，下图右侧的 Phenix.TeamHub.Prober.dll 就是 Phenix丶为采集缺陷而提供的开放性框架程序集，左侧的 Phenix.TeamHub.Prober.Business.dll，是可以注册到这框架上的一个具体实现日志处理功能的样例模块：



Phenix.TeamHub.Prober.AppHub 静态类的 Worker 属性，可接驳（/可注册/被赋值）上任何符合 Phenix.TeamHub.Prober.IWorker 接口要求的第三方（开发的）操作者对象。

Phenix.TeamHub.Prober.IWorker 接口定义如下：

```
/// <summary>
/// 操作者
/// </summary>
public interface IWorker
{
  #region 方法

  /// <summary>
  /// 提交日志
  /// </summary>
  /// <param name="message">消息</param>
```

```
    /// <param name="method">函数的信息</param>
    /// <param name="exception">错误</param>
    void SubmitLog(string message, MethodBase method, Exception exception);

    #endregion
  }
```

实现 Phenix.TeamHub.Prober.IWorker 接口的操作者对象，应该能处理到这些传入的参数，作为日志信息来持久化。这样，Phenix.TeamHub.Prober.AppHub 可以作为统一的日志处理入口，被编织到业务系统任何需要记录日志的程序集函数里。

当然，你也可以不使用 Phenix 在 IDE 里提供的 Addin 静态编织工具（下文介绍用法），你可以直接引用 Phenix.TeamHub.Prober.dll 程序集，然后在需要记录日志的函数里直接调用 Phenix.TeamHub.Prober.AppHub 的静态函数，并用“#if 条件编译符号”-“#endif”圈注。这样，可达到一样的效果，但缺点也很明显，业务代码里混杂着大量日志记录代码，侵略性强：

```
    /// <summary>
    /// 提交日志
    /// </summary>
    /// <param name="method">函数的信息</param>
    /// <param name="exception">错误</param>
    public static void SubmitLog(MethodBase method, Exception exception)

    /// <summary>
    /// 提交日志
    /// </summary>
    /// <param name="message">消息</param>
    /// <param name="method">函数的信息</param>
    /// <param name="exception">错误</param>
    public static void SubmitLog(string message, MethodBase method, Exception exception)
```

2.1.2 实现 IWorker 接口

在 Phenix.Extensions 目录里提供了 Phenix.TeamHub.Prober.Business 工程，作为实现 IWorker 接口的示例代码，可供参考开发出完全脱离 Phenix.Net 框架的日志处理模块。

```
  /// <summary>
  /// 操作者
  /// </summary>
  public class Worker : Phenix.TeamHub.Prober.IWorker
  {
    /// <summary>
```

```
    /// 注册
    /// </summary>
    public static void Register()
    {
      Phenix.TeamHub.Prober.AppHub.Worker = new Worker();
    }

    /// <summary>
    /// 提交日志
    /// </summary>
    void Phenix.TeamHub.Prober.IWorker.SubmitLog(string message, System.Reflection.MethodBase method,
Exception exception)
    {
      new SubmitLogCommand(message, method, exception).Execute();
    }
  }
}
```

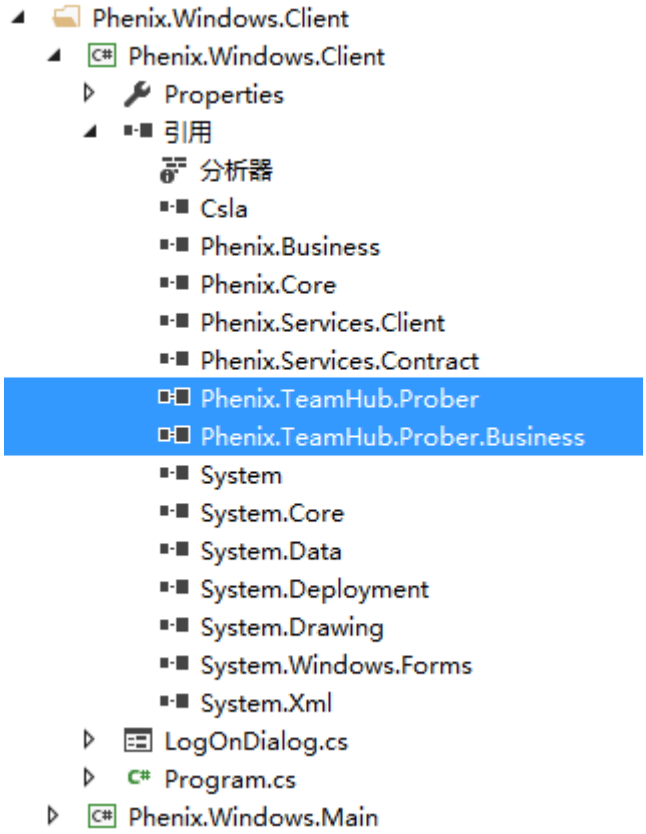
存储日志的核心功能写在了 Phenix.TeamHub.Prober.Business.SubmitLogCommand 类里，日志被持久化到以下的表中：

```
CREATE TABLE PT_ExecuteLog (                  --执行日志
  EL_ID NUMERIC(15) NOT NULL,
  EL_Time DATE NOT NULL,                      --时间
  EL_UserID VARCHAR(100) NULL,                --用户ID
  EL_UserNumber VARCHAR(100) NULL,            --用户工号
  EL_Message VARCHAR(4000) NULL,              --消息
  EL_AssemblyName VARCHAR(255) NULL,          --程序集名
  EL_NamespaceName VARCHAR(255) NULL,         --命名空间名
  EL_ClassName VARCHAR(255) NULL,             --类名
  EL_MethodName VARCHAR(255) NULL,            --方法名
  EL_ExceptionName VARCHAR(255) NULL,         --错误名
  EL_ExceptionMessage VARCHAR(4000) NULL,     --错误消息
  EL_ExceptionStackTrace LONG /*TEXT*/ NULL, --错误调用堆栈
  PRIMARY KEY(EL_ID)
)
```

如果希望使用全套的 Phenix Teamwork Tools 功能，请不要改变这个表结构，即使是在你自己开发的日志处理模块里，也请把日志持久化到这个表里。如果你仅是拿来主义，以上内容并不值得关心，接下来的才是正文。

2.1.3 注册日志处理

要能采集到业务系统的缺陷，首先在其主程序里引用到 Phenix.TeamHub.Prober.dll、Phenix.TeamHub.Prober.Business（可以是你自己的日志处理模块）程序集，具体方法参考 Phenix.Windows.Client 工程：



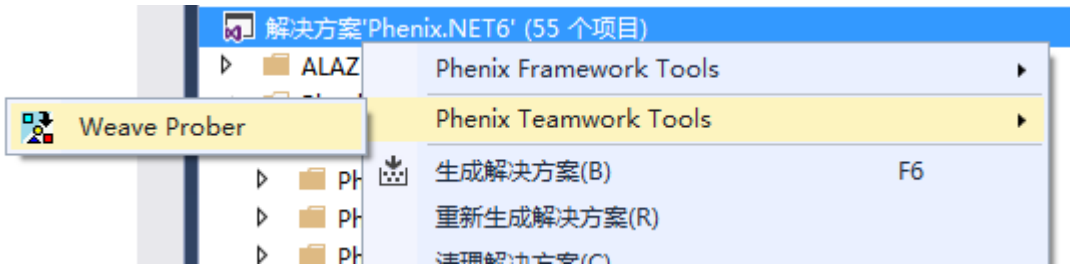
然后在主程序里嵌入如下代码：

```
if (user != null && user.Identity.IsAuthenticated)
{
  //Phenix.Core.AppConfig.AutoStoreLayout = true; //允许自动保存界面Layout
  //Phenix.Core.Net.NetConfig.RegisterServicesCluster("本地服务", "127.0.0.1"); //可注册服务集群
  Phenix.TeamHub.Prober.Business.Worker.Register(); //注册Phenix.TeamHub.Prober.AppHub的功能实现
  //启动主Form
  PluginHost.Default.SendSingletonMessage("Phenix.Windows.Main", null);
   }
```

业务系统改动代码的地方就这一行，这是应框架的开放性所必须做的。

2.1.4 编织日志刺针

所有业务系统的功能模块，都可以通过 Phenix丶在 IDE 里提供的 Addin 工具来静态编织进日志刺针代码：

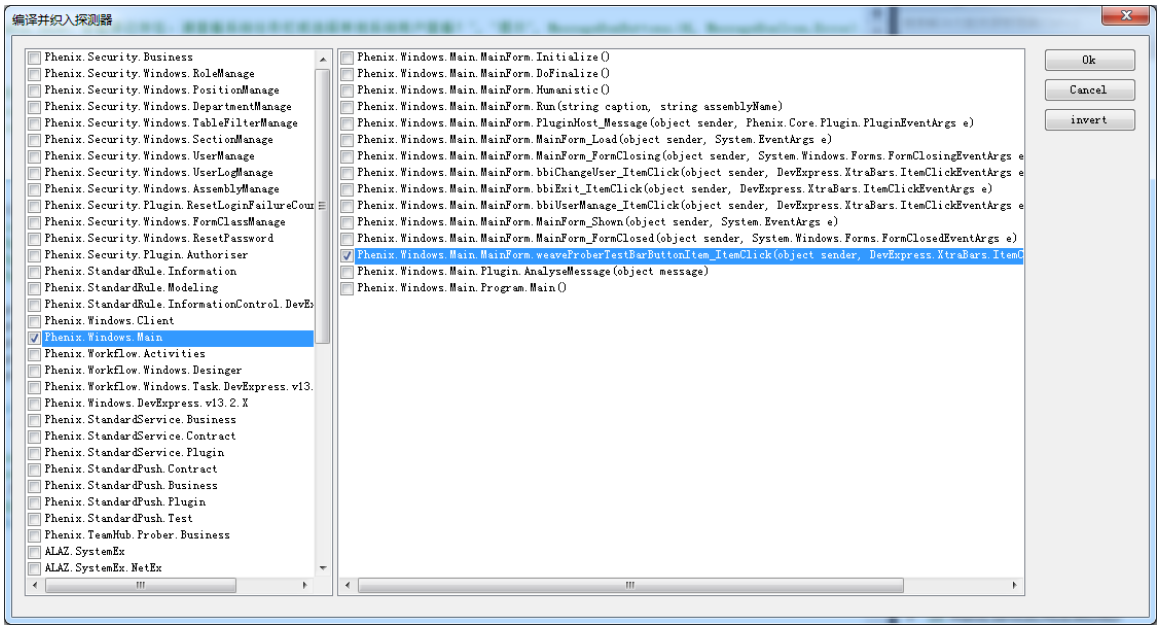


为讲解用法，在 Phenix.Windows.Main 工程里提供了测试代码：

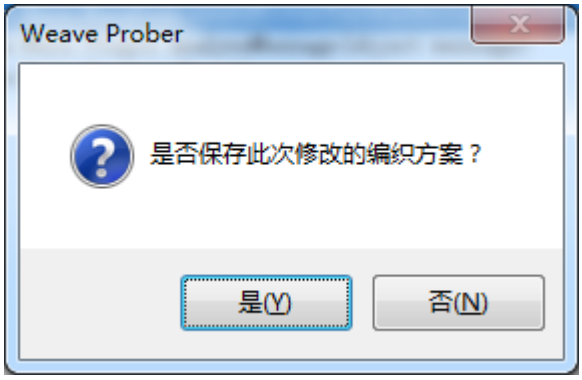

```
private void weaveProberTestBarButtonItem_ItemClick(object sender, ItemClickEventArgs e)
{
  ////测试：故意抛出被零除的缺陷以测试织入探针功能
  //请事先通过IDE的Phenix Teamwork Tools的Weave Prober将探针织入本函数
  int i = 1;
  i = i / 0;
 }
```

显然，这段代码是有缺陷的，为的是测试缺陷采集功能是否有效。

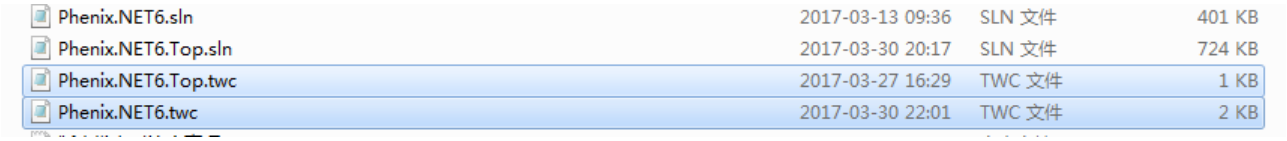
在 IDE 的解决方案、工程、编辑界面上，都可以按右键点选 Weave Prober 功能按钮来弹出以下的界面：



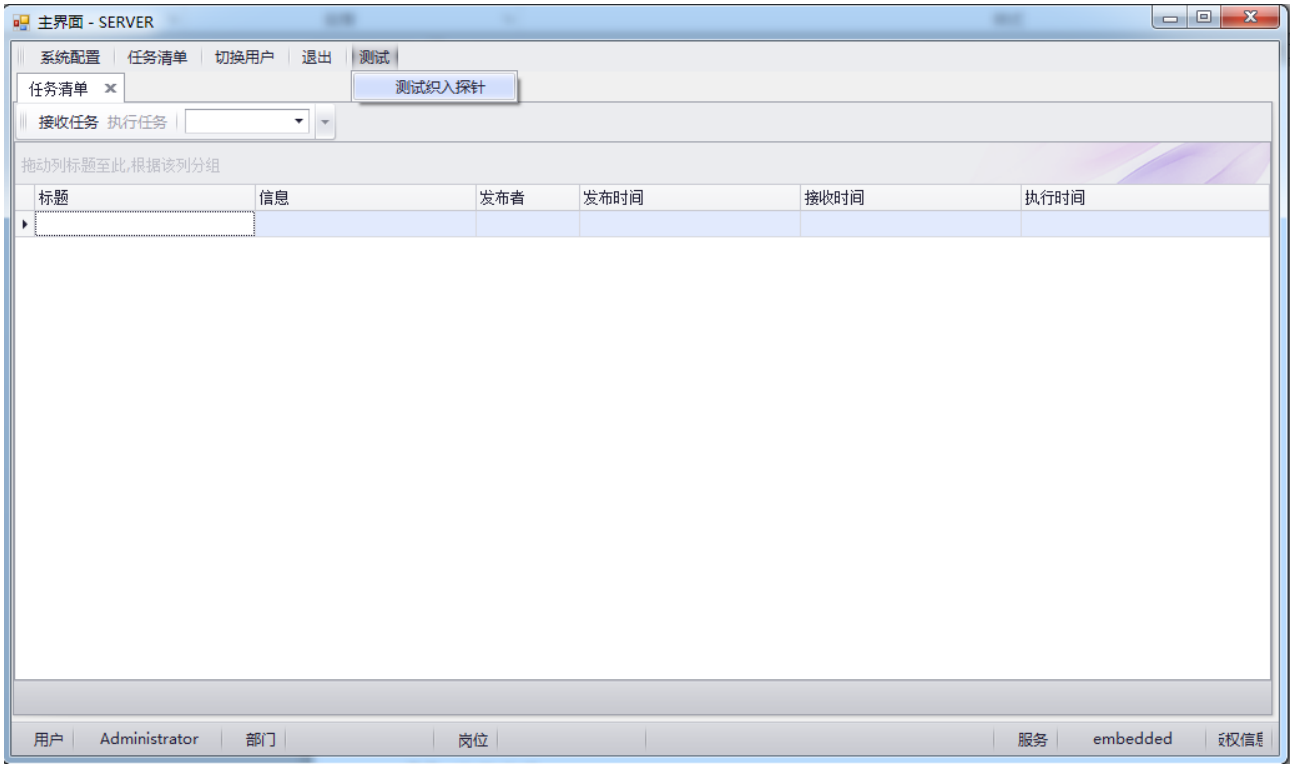
完成编织后，可以将编织方案保存到解决方案所在目录下，以便下次继续使用。



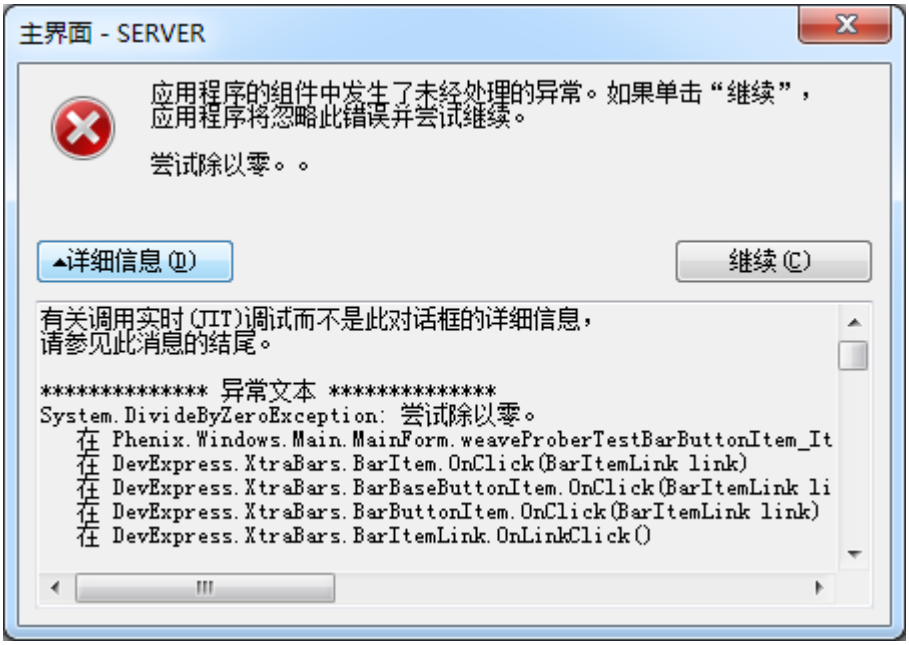
编织方案的文件名与解决方案相同，后缀：



接下来，我们执行 Phenix.Windows.Client.exe 程序进入到主界面开始测试：

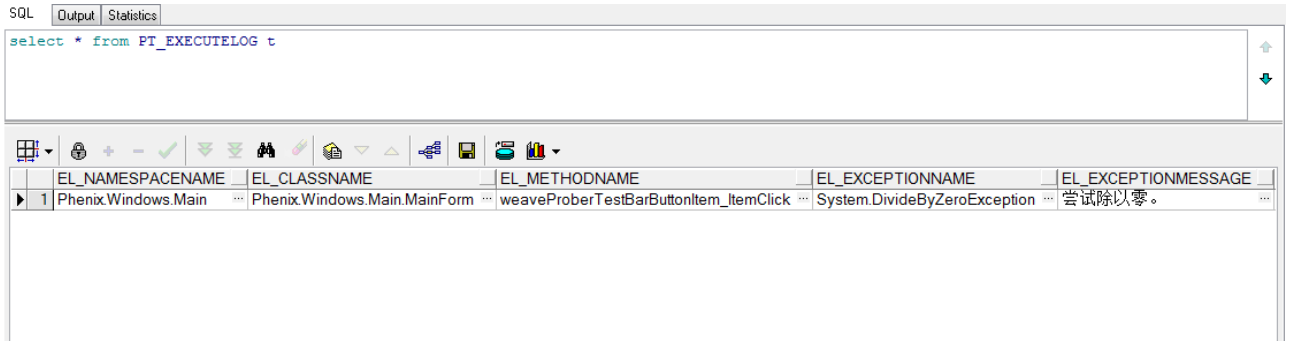


只要点击这主界面上的“测试织入探针”功能菜单，就会弹出错误消息：



说明日志刺针并未吞吃掉异常事件。

此时，异常事件应该已经被日志刺针保存到了 PT_ExecuteLog 表中，查看下果然新增了一条记录：



2.2 缺陷处理

采集到的每个缺陷，需确保被及时解决。

待续。

2.3 缺陷跟踪

待续。

2.4 缺陷统计

待续。