MiniProfie 在单元测试中的应用

1. 配置 log4net 日志组件

   略

2. 单元测试类中，指定监视器 provider，指定 sql 格式化器

```
        /// 初始化测试环境
        /// 相当于Application_Start
        /// </summary>
        [TestFixtureSetUp]
        public void TestFixtureSetUp()
        {
            //生成日志文件
            var path = AppDomain.CurrentDomain.SetupInformation.ApplicationBase +
                       "\\config\\log4net_local.config";
            var fi = new System.IO.FileInfo(path);
            log4net.Config.XmlConfigurator.Configure(fi);

            //指定provider
            MiniProfiler.Settings.ProfilerProvider = new SingletonProfilerProvider();
            //指定sql格式化器，输出带实参sql
            MiniProfiler.Settings.SqlFormatter = new
StackExchange.Profiling.SqlFormatters.InlineFormatter();
            MiniProfiler.Settings.
        }

        /// <summary>
```

   初始化日志组件

   指定 provider 类型为单例 SingletonProfilerProvider，本模式适用于大多数 unit test。

   指定 sqlformatter 为 Inlineformater，输出带实参的 sql 语句。

3. 对 IdbConnection 进行包装，并返回包装过的连接

```
    public static DbConnection GetMySqlConnection()
    {
        var connection = new MySqlConnection(ConfigurationManager.ConnectionStrings["huatongdb"].ConnectionString);
        //connection.Open();
        //return connection;
        var profiledConn= new StackExchange.Profiling.Data.ProfiledDbConnection(connection, MiniProfiler.Start());
        profiledConn.Open();
        return profiledConn;
    }
    #endregion

    "封装dapperExtensions扩展操作方法"
```

   用监视类对 connection 进行包装，注意第二个参数，本处

在 SingletonProfilerProvider 的基础上创建一个 Miniprofile 监视器，并置为当前监视器，可通过 MiniProfile.Current 静态属性直接访问。

4. 编写单元测试方法，插入监视操作语句

```
[Test]
public void GetTicketLineOrderByOrderNumOverWriteTest()
{
    ITicketLineOrderService service = new TicketLineOrderService();
    MiniProfiler.Start();//因为是单例，所以本行代码可省略
    var order = service.GetTicketLineOrderByOrderNumOverWrite ("2b758401400c0800");
    MiniProfiler.Stop();
    //将捕捉到的所有监视信息直接式化为Json
    //包括sql，http请求等监视信息
    var timingInfo= MiniProfiler.Current.Root.CustomTimingsJson;
    //输出到日志文件
    LogHelper.WriteLog(timingInfo);
    //输出到控制台
    System.Console.WriteLine(timingInfo);
    //也可如下只取sql有关的监视信息，key必须填写为sql
    //然后手动序列化
    //var sqlInfo = MiniProfiler.Current.Root.CustomTimings["sql"];
    //var sqlInfoStr = new JavascriptSerializer().Serialize(sqlInfo);
    Assert.IsNotNull(order);
}
```

1. 取得当前监视器，开始监视，因为是单例模式的 provider，本行代码可省略
2. 执行业务方法的单元测试后，停止监视时间记录
3. 读取监视记录信息，返回信息格式为 Json 字符串
4. 输出到 log 日志
5. 输出到单元测试控制台

控制台输出

```
测试启动
{"sql":[{"Id":"916a3e25-622b-403b-84f2-715f20df2b9c","CommandString":"select * from tb_ticketlineorder where OrderNum = @orderNum","ExecuteType":"Reader","StackTraceSnippet":"GetTicketLineOrderByOrderNumOverWrite GetTicketLineO
测试结束

1 passed, 0 failed, 0 skipped, took 22.43 seconds (NUnit 2.4).
```

日志输出

```
"sql": [{
    "Id": "6fedbad8-fb04-4117-ae90-f37f624dcd28",
    "CommandString": "select * from tb_ticketlineorder where OrderNum = \u00272b758401400c0800\u0027",
    "ExecuteType": "Reader",
    "StackTraceSnippet": "GetTicketLineOrderByOrderNumOverWrite GetTicketLineOrderByOrderNumOverWriteTest InvokeMethod InvokeMethod RunT
    "StartMilliseconds": 2010.9,
    "DurationMilliseconds": 586.4,
    "FirstFetchDurationMilliseconds": 545.6
},
{
    "Id": "0fdc9257-9b9d-4a53-8485-c1bf620649c5",
    "CommandString": "select * from tb_orderticket where order_id =\u0027000179dc-183d-4c7e-b5fc-4cec8b84abd3\u0027",
    "ExecuteType": "Reader",
    "StackTraceSnippet": "GetTicketLineOrderByOrderNumOverWrite GetTicketLineOrderByOrderNumOverWriteTest InvokeMethod InvokeMethod RunT
    "StartMilliseconds": 2602,
    "DurationMilliseconds": 10.5,
    "FirstFetchDurationMilliseconds": 2.3
}]
```

监视的信息类型为 sql 语句，

Sql 语句内容。

语句送出到结果返回的持续时间

带实参的 sql 语句输出是带有转移符\u0027，就是单引号。

如果有业务方法内有多个数据库操作，甚至这终有有缓存数据的读取，比如 redis，memcache 的访问，其访问信息也会在此处输出。

类型会标识为“redis”或者“memcache”等，而不是 sql。

Tips:因单元测试中# IF Debug 预处理命令在 debug 模式下不能被识别。在测试完毕后，需要手动修改 IDbConnection 为未包装类型，否则会影响到程序发布后的程序性能

```
public static DbConnection GetMySqlConnection()
{
    var connection = new MySqlConnection(ConfigurationManager.ConnectionStrings["huatongdb"].ConnectionString);
    connection.Open();
    return connection;

    //var profiledConn= new StackExchange.Profiling.Data.ProfiledDbConnection(connection, MiniProfiler.Start());
    //profiledConn.Open();
    //return profiledConn;
}
#endregion
```

2014-8-5