# NHibernate

NHibernate简介

什么是？NHibernate？NHibernate是一个面向.NET环境的对象/关系数据库映射工具。对象/关系数据库映射（object /relational mapping，ORM）这个术语表示一种技术，用来把对象模型表示的对象映射到基于SQL的关系模型数据结构中去。

在今日的企业环境中，把面向对象的软件和关系数据库一起使用可能是相当麻烦和浪费时间的。而NHibernate不仅仅管理.NET类到数据库表的映射（包括.NET 数据类型到SQL数据类型的映射），还提供数据查询和获取数据的方法，可以大幅度减少开发时人工使用SQL和ADO.NET处理数据的时间。

NHibernate的目标主要是用于与数据持久化相关的编程任务，能够使开发人员从原来枯燥的SQL语句的编写中解放出来，解放出来的精力可以让开发人员投入到业务逻辑的实现上。对于以数据为中心的程序，开发人员往往是在数据库中使用存储过程来实现商业逻辑，这种情况下NHibernate可 能不是最好的解决方案，但对于那些基于.NET，并且能够实现OO业务模型和商业逻辑的中间层应用，NHibernate是最有用的。 NHibernate可以帮助用户消除或者包装那些针对特定厂商的SQL代码，并且帮用户把结果集从表格式的表示形式转换成一系列的对象。

NHibernate是一个目前应用的最广泛的开放源代码的对象关系映射框架，它对Java的JDBC（类似于ADO.Net）进行了非常轻量 级的对象封装，使得程序员可以随心所欲的使用对象编程思维来操纵数据库，目前在国内Java开发界已经颇为流行，Hibernate + Spring往往是 很多Java公司招聘的要求。而 NHibernate，顾名思义，如同NUnit，NAnt一样，是基于.Net的Hibernate实现，但是目前介绍 NHibernate的资料非常少，缺少一个系统完整的教程来全面的展现和深入NHibernate，而且现在NHibernate的文档又残缺不全，少 的可怜，很多NHibernate的学习者往往都是通过Hibernate的文档来学习，但是毕竟不是所有的.Net开发者都熟悉Java，也不是所有的 人都有精力有时间去学习Java，所以，我准备开始一个Step by Step的NHibernate教程，以便有兴趣的 朋友能够快速的熟悉NHibernate，能够更快地体验NHibernate的开发乐趣。 NHibernate 是一个基于.Net 的针对关系型数据库的对象持久化类库。 NHibernate 来源于非常优秀的基于Java的Hibernate 关系型持久化工具。 NHibernate 从数据库底层来持久化你的.Net 对象到关系型数据库。 NHibernate 为你处理这些，远胜于你不得不写SQL去从数据库存取对象。你的代码仅仅和对象关联，NHibernate 自动产生SQL语句，并确保对象提交到正确的表和字段中去。

NHiberante的优缺点

* **优点**

(1).面向对象：NHiberante的使用时只需要操纵对象，使开发更对象化，抛弃了数据库中心的思想，完全的面向对象思想。

(2).透明持久化：带有持久化状态的、具有业务功能的单线程对象，此对象生存期很短。这些对象可能是普通的POCO，这个对象没有实现第三方 框架或者接口，唯一特殊的是他们正与（仅仅一个）Session相关联。一旦这个Session被关闭，这些对象就会脱离持久化状态，这样就可被应用程序 的任何层自由使用。（例如，用作跟表示层打交道的数据传输对象。） (3).它没有侵入性，即所谓的轻量级框架。正因为它具有透明持久化的优点，它才没有侵入性，才是一个轻量级框架。恒定一个框架为重量级、还是轻量级，是根据其侵入性而定夺的。而NHibernate就是一个轻量级ORM框架。

(4).较好的移植性：支持多种数据库，便于数据库的迁移。

(5).缓存机制：提供一、二级缓存和查询缓存。

(6).开发效率：众所周知，使用NHibernate可以简化程序开发，从而达到快速开发的目的。作为软件公司，项目管理的关键就是控制开发 成本。正因为使用NHibernate后所写的代码量减少了，相对于原先使用“SqlHelper、DAL、BLL”开发程序的项目周期缩短了，成本就降 低了。

* **缺点**

(1).内存消耗：直接使用“SqlHelper、DAL、BLL”无疑是最省内存的。使用NHibernate后，内存开销比较大，这点是毋庸置疑的。

(2).批量数据库的处理：由于NHibernate是基于面向对象的ORM框架，处理数据库的方式是针对单个对象的。对数据库的增、删，改都是正对一条记录而言。对于批量修改、删除数据，不适合用NHiberante。这也是所有OR框架弱点，其原因，我认为是在于与缓存机制的冲突。

(3).较多使用数据库特性时，也不适合使用NHiberante。如数据库中大量的存储过程、触发器、特点的SQL语句。

(4).表关系比较混乱时也不适合使用NHiberante。NHibernate只适合于表与表的关系比较明确的环境中。如本应该建立外键的，没有建立外键。这时使用NHiberante不仅没有减少工作量，反而增加了工作量。

(5).学习成本：相对于NHibernate来说，使用“SqlHelper、DAL、BLL”操作数据库，学习成本比较低，而且上手很快。 使用NHibernate需要有一定OOP(面向对象编程)和OOD(面向对象设计)的基础，这对于基础薄弱的程序员来说，从面向过程的编程到面向对象的 编程，需要一定的投入；一般情况下需要学习1个月左右的时间才能够深入NHiberante。

NHibernate的应用

详见demo程序。

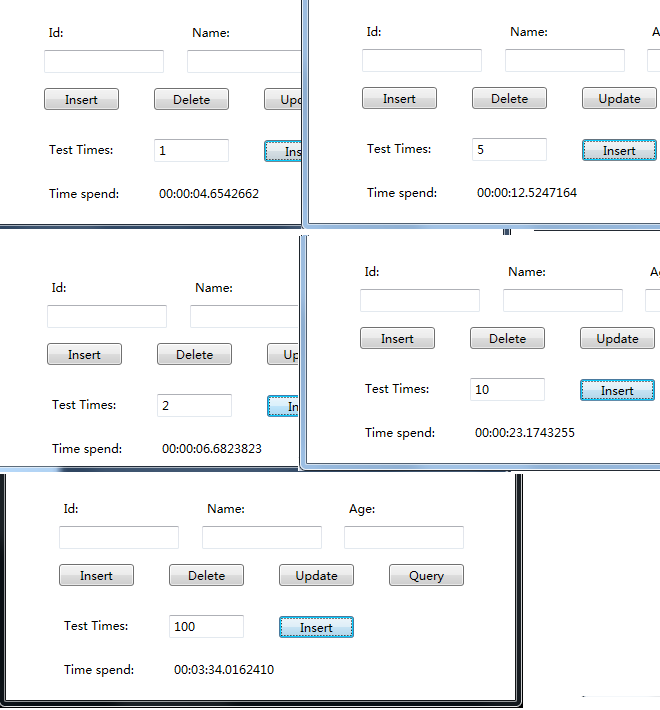
参考文档：<http://wenku.baidu.com/link?url=mqrz7kaPIA_nZrmqbWEUMZK8Bsr7-hszxONLlpvLWag3IuD8T68c1woBmRZfyJmZJb0hMrkhlXUWLFwxd_W3KK2q9wXE08XvWmCuqNWilhq> 全面详细的说明《NHibernate从入门到精通》

<http://nhibernate.info/doc/tutorials/first-nh-app/your-first-nhibernate-based-application.html> 官方first application教程

<http://blog.csdn.net/sunqifeng/article/details/5034404> 以及官方教程的双语翻译版

NHibernate的性能测试

由于NHibernate的缺点（1）（2），做了一些针对性的测试，界面上提供了批量插入的操作（循环调用NHibernate的插入操作），分别进行了1次，2次，5次，10次以及100次的插入操作，单词操作的数据大小为1K，测试结果如下：



可以发现NHibernate在做批量操作时的局限性。