**Starting NHibernate (A beginners guide).**

**Introduction:**

The purpose of this article is to provide a basic introduction and guideline to start working with NHibernate. We will discuss the required technique and concepts first and later part of this article we will create a practical projects.

NHibernate is an ORM (Object-Relational mapper) for .net platform just like Entity Framework. But it is an open source tool supported by NHibernate community. Official links to NHibernate site is <http://nhibernate.info/>

If you do not know what ORM is please read following links:

<https://en.wikipedia.org/wiki/Object-relational_mapping>

<http://stackoverflow.com/questions/959270/what-is-nhibernate>

**How to use**

To use NHebernate into your application you have to perform following three steps

1. Install and add references of NHibernate .dlls to the application
2. Configure NHibernate so that you can create session factory and session
3. Map your database tables with your application objects (class)

Let us discuss each of the steps in detail. We are just discussing the requirements and techniques here; we will try to create a c# project at the end of the article.

**Install NHibernate**

You can easily download the latest version of **NHibernate** from their official site. Just unzip the package and add reference of the required dlls. Normally you will require “NHibernate.dll” and “Iesi.Collections.dll”. The best approach and you should use the NuGet package manager to download and install NHibernate.

**NHibernate.Cfg.Configuration**

The instance of “NHibernate.Cfg.Configuration” class holds all the mapping between .net objects and database objects (we will discuss later how to provide the mapping to the configuration). The configuration instance uses various configuration parameters. NHibernate provides a lots of configuration parameters to support many different environments. All the configuration parameter has their optional value. But we can also customize the configuration value according to our needs.

Creating instance is as simple as creating instances of any of the .net classes:

var cfg = new NHibernate.Cfg.Configuration();

or if we use “NHibernate.Cfg” at using directives then simply

var cfg = new Configuration();

**Configure NHibernate**

We can provide custom configuration value using

1. App/web config file: We have to add a separate configsection of type “NHibernate.Cfg.ConfigurationSectionHandler” to provide the configuration parameter. Following is an example

﻿<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<configSections>

<section name="hibernate-configuration" type="NHibernate.Cfg.ConfigurationSectionHandler, NHibernate" />

</configSections>

<hibernate-configuration xmlns="urn:nhibernate-configuration-2.2">

<session-factory>

<property name="dialect">NHibernate.Dialect.MsSql2012Dialect</property>

<property name="connection.connection\_string\_name">DBConnection</property>

<mapping assembly="LearningNH" />

</session-factory>

</hibernate-configuration>

<connectionStrings>

<add name="DBConnection" connectionString="Data Source=BULBUL\MSSQLSERVER12;Initial Catalog=EmployeeDB;Trusted\_Connection=True;"/>

</connectionStrings>

</configuration>

1. XML configuration file: We use a separate xml file named “hibernate.cfg.xml” to specify the configuration parameter. This file should be resided to the application directory otherwise we need to specify the file path at the code. Following is an example

<?xml version='1.0' encoding='utf-8'?>

<hibernate-configuration xmlns="urn:nhibernate-configuration-2.2">

<!-- an ISessionFactory instance -->

<session-factory>

<!-- properties -->

<property name="connection.provider">NHibernate.Connection.DriverConnectionProvider</property>

<property name="connection.driver\_class">NHibernate.Driver.SqlClientDriver</property>

<property name="connection.connection\_string">Server=localhost;initial catalog=nhibernate;User Id=;Password=</property>

<property name="show\_sql">false</property>

<property name="dialect">NHibernate.Dialect.MsSql2000Dialect</property>

<!-- mapping files -->

<mapping resource="NHibernate.Auction.Item.hbm.xml" assembly="NHibernate.Auction" />

<mapping resource="NHibernate.Auction.Bid.hbm.xml" assembly="NHibernate.Auction" />

</session-factory>

</hibernate-configuration>

1. Programmatically : We can set configuration properties using “SetProperty” method of “Configuration” instance. This method takes name and value of the configuration as parameter. For example

cfg.SetProperty(NHibernate.Cfg.Environment.ShowSql, "true");

cfg.SetProperty(NHibernate.Cfg.Environment.FormatSql, "false");

cfg.SetProperty(NHibernate.Cfg.Environment.Dialect, "NHibernate.Dialect.MsSql2012Dialect");

var connectionString = System.Configuration.ConfigurationManager.ConnectionStrings[“DbConnection”]

cfg.SetProperty(NHibernate.Cfg.Environment.ConnectionString, connectionString);

**Adding mapping files to the configuration**

Before creating IsessionFactory we need to add the mapping files to the configuration. How to create mapping file, we will discuss later but for now be informed that the mapping file is an xml file with extension “.hbm.xml”

There are various ways to add the mapping files like,

1. cfg.AddFile("Item.hbm.xml")
2. cfg.AddClass(typeof(Item))//Item is a .net class type
3. cfg.AddInputStream(memorystream)// memorystream is a stream of mapping file
4. cfg.AddAssembly("LearningNH")// LearningNH is the name of the assembly

Probably the best option is to add the assembly. In this case we need to set the “build action” property of the mapping files to “Embedded Resources”. NHibernate will find the mapping files from the embedded resources.

**ISessionFactory**

We need an instance of “NHibernate.Cfg.Configuration” class to create “IsessionFactory”.