# NHibernate

### Object-Relational Mapper

NHibernate is a port of the Java Hibernate object relational mapper (ORM). ORMs translate your classes and objects into SQL commands for storing those objects and retrieving them from a relational database. Gentle.NET is another such ORM for .NET and Mono.

### Code Sample

By Hans-Christian Holm (see this note).

The DLLs from nHibernate and the PostgreSQL .NET provider (Npgsql) are needed to run this sample.

---

The example uses a PostgreSQL server on localhost with a "person" table in the "test" database, but should work with any common database system.

nHibernate examples often use the mapping files (*.hbm.xml) as embedded resources, typically along with the business objects in a separate DLL, but I haven't figured out how to do that with booc.exe. Specifying the mapping file as a resource does not work with booc.exe.

In nHibernate, there is a contributed library (net2hbm) that allows you to use attributes to declare mappings instead of xml files, but it's rather new and undocumented right now. It's based on the similar attributing technique in (Java) Hibernate.

BooHibTest.boo:

**BooHibTest.boo**

```
namespace BooHibTest

import System.Text.RegularExpressions
import NHibernate

# business object
class Person:
  # these properties map to database fields
  [property(ID)]    _id as int
  [property(Name)]  _name as string

  # "business method" that return initials
```

```
in name
  Initials as string:
    get:
      matches = /(?<!\w)\w/.Matches(_name)
      return join(matches, "")

# initialise nHibernate stuff
cfg = Cfg.Configuration()
cfg.AddXmlFile("BooHibTest.hbm.xml")
fact as NHibernate.ISessionFactory =
cfg.BuildSessionFactory()
sess as ISession = fact.OpenSession()

# add "-san" to all names with an "x" in
them
xnames = sess.Find("from Person where Name
like '%x%'")
for p as Person in xnames:
  p.Name += "-san"
  print "$(p.Name) ($(p.Initials))"
```

```
# update DB, close
sess.Flush()
sess.Close()
```

BooHibTest.hbm.xml:

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping
xmlns="urn:nhibernate-mapping-2.0">

 <class name="BooHibTest.Person, BooHibTest"
table="person">
  <id name="ID" column="id" type="Int32">
   <generator class="assigned" />
  </id>
  <property name="Name" column= "name"
type="String" length="100"/>
 </class>

</hibernate-mapping>
```

BooHibTest,exe,config:

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

  <configSections>
    <section name="nhibernate"

type="System.Configuration.NameValueSectionH
andler, System,
```

```
         Version=1.0.5000.0,Culture=neutral,
PublicKeyToken=b77a5c561934e089"
    />
    <section name="log4net"

type="log4net.Config.Log4NetConfigurationSec
tionHandler, log4net"
    />
 </configSections>

 <log4net>
  <!-- whatever suits your needs here -->
 </log4net>

  <nhibernate>
    <add
      key="hibernate.connection.provider"

value="NHibernate.Connection.DriverConnectio
nProvider"
    />
    <add
      key="hibernate.dialect"

value="NHibernate.Dialect.PostgreSQLDialect"
    />
    <add

key="hibernate.connection.driver_class"
      value="NHibernate.Driver.NpgsqlDriver"
    />
```

```
    <add

key="hibernate.connection.connection_string"

value="Server=localhost;Database=test;User
Name=postgres;Password=postgres"
    />
```

```
  </nhibernate>

</configuration>
```

Compile and run:

```
booc -r:NHibernate.dll BooHibTest.boo
./BooHibTest.exe
```

## See Also

Gentle.NET ORM and other Database Recipes.