



NHibernate

By
Philip Japikse
Phil.japikse@pinnsng.com
MVP, MCSD.NET, MCDBA, CSM, CSP
Principal Consultant
Pinnacle Solutions Group



Who am I?

- ▶ Principal Consultant, Pinnacle Solutions Group
- ▶ Microsoft MVP
- ▶ MCSD, MCDBA, CSM, CSP
- ▶ Enterprise Application Architect
- ▶ Trainer/Mentor/Speaker
- ▶ Lead Director, Cincinnati .NET User's Group
- ▶ Contributing Author – www.nplus1.org

Motivation

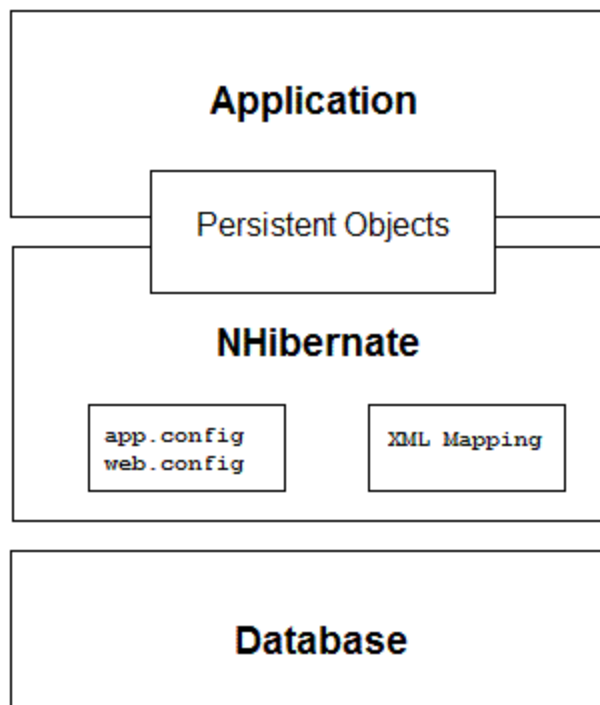
▶ CRUD SUCKS

Agenda

- ▶ Architecture
- ▶ Data Objects
- ▶ Configuration
- ▶ Mapping
- ▶ Collections
- ▶ Sessions and Transactions
- ▶ Querying
- ▶ Manipulating Persistent Objects
- ▶ Real World Customizations
 - Custom Connection Provider
 - SQL Server Timestamp for Concurrency
- ▶ Additional Resources

Architecture

▶ Architecture



Data Objects

- ▶ Goal to be DAL implementation agnostic
 - Not always attainable
- ▶ POCO exist in one of three states:
 - Transient
 - Don't exist in the data store
 - Persistent
 - Exist in the data store
 - Detached
 - Persistent objects no longer associated with the persistence manager

Data Objects (cont)

▶ Four Main Rules:

- Declare getters/setters for all persistent properties
 - Can use auto props if not needing INotifyPropertyChanged
 - Do NOT need to be public
 - DO need to match mapping
- Must have no-arg constructor for NHibernate's use of Activator.CreateInstance
 - Does NOT have to be public
- Provide an Identifier field
 - Maps to the <id> element in the mapping document
- Class s/b non-sealed and all public methods declared as virtual
 - Enables NHibernate to wrap pocos with proxies
 - Used in performance tuning among other "magic"

Interfaces

- ▶ NHibernate
 - ILifeCycle – deprecated
 - Provides callbacks for vetoing Save/Update/Delete operations on an object
- ▶ .NET (My Implementations)
 - INotifyPropertyChanged
 - Used in data binding scenarios
 - IComparable
 - Allows adding business rules around sorting
 - IDataErrorInfo
 - Mainly used in WPF applications

Configuration

- ▶ XML Configuration is either done through:
 - *.config (better choice)
 - hibernate.cfg.xml
- ▶ Configuration section
 - name="hibernate-configuration"
 - type="NHibernation.Cfg.ConfigurationSectionHandler, NHibernate"
- ▶ Loaded with NHibernate.Cfg.Configuration
 - If no file specified, looks in *.config file, then looks for hibernate.cgf.xml

Configuration Elements

- ▶ Key Properties for <session-factory>:
 - “connection-provider” – handles connections
 - Example shows custom provider
 - “dialect” – target database for SQL dialects
 - Example shows custom dialect
 - “show_sql” – outputs sql calls to windows
 - Useful for debugging purposes
 - TURN OFF IN PRODUCTION
 - “mapping” – assembly that contains the mapping files
 - Otherwise must add the files into the configuration through code

Mapping

- ▶ Define the bridge between persistent POCO (Plain Old CLR Objects) and Database
- ▶ Three main paradigms:
 - Table per Concrete Class
 - Table per Class Hierarchy
 - Table per Subclass

Mapping

- ▶ Main mapping methods:
 - XML Documents (ClassName.hbm.xml)
 - I prefer code generation of separate XML mapping and POCO files
 - Keeps POCO data access agnostic
 - Attributes on POCOs
 - Castle ActiveRecord and other tools build on this and offer additional methods

Mapping

- ▶ When Using XML Documents:
 - Root element must be `<hibernate-mapping>`
 - Files must be set as embedded resource for Project
 - Each persistent class/Interface is defined by a `<class>` element
 - More than one class per file is allowed, considered bad form
 - Objects (and collections) can be defined for Lazy Load
 - Overridable at query time

Mapping – Concurrency

- Concurrency columns (incremented on each change):
 - <version> – Int32
 - <timestamp> – DateTime (not SQL Server timestamp data type)
 - Non-NHibernate changes will cause collisions
 - Should use SQL Server Timestamp column (requires custom implementation)

Collections

- ▶ Three basic types of collections:
 - Set = Unique items, no order
 - Bag = no order
 - List = ordered, each item has an index
- ▶ Inverse allows for bidirectional associations
- ▶ Lazy loading or Not?
 - Lazy is better choice, but must initialize items prior to use
- ▶ Cascade
 - All | none | save-update | delete | all-delete-orphan
- ▶ Order-by
- ▶ Batch-size

Handling Collections

- ▶ When One-To-Many collections are lazy loaded, they must be initialized prior to being handled
 - NHibernate.IsInitialized
 - NHibernate.Initialize
 - If Parent is in Detached state, must use session.Lock
- ▶ When creating new objects with bidirectional collections, must code both sides:

```
Child c = new Child();  
Parent p = new Parent();  
c.Parent = p;  
p.Children.Add(c);
```

Sessions and Transactions

- ▶ Sessions are analogous to connections
 - One ISessionFactory per database
 - Created via ISessionFactory.OpenSession()
- ▶ Recommend to use transaction
 - Eliminates need to call Session.Flush()
- ▶ Sample usage:

```
using (ISession session = SessionFactory.OpenSession()) {  
    try {  
        using (ITransaction trans = session.BeginTransaction()) {  
            try { session.Update(entity); }  
            catch (Exception ex) { transaction.Rollback(); }  
        }  
    }  
    finally { SessionFactory.CloseSession(); }  
}
```

Sessions and Transactions

- ▶ Always call `ISession.Close()` when app exits
 - Implement `IDisposable` on `SessionManager`
- ▶ **Interceptor**
 - Much more robust than `ILifeCycle`, can be used for auditing and other tasks

Querying

- ▶ **Single POCO query**
 - If you know the primary key value
 - Get<T> – loads object from the database
 - Returns null if not found
 - Load<T> – loads the object from the cache or the database
 - Returns proxy if not found, throws exception when accessed
- ▶ **IList<T> queries**
 - IQuery – Hibernate Query Language
 - Similar to LINQ syntactically
 - ISQLQuery – Native SQL
 - Based on Dialect
 - ICriteria – OO implementation
 - Expressions can used to refine the search
 - Stored Procedures

Projections

- ▶ Can use Projection to create instance of different object based on query
 - Similar to LINQ to Objects
 - See sample code or this: <http://tinyurl.com/dmc8cy>

Managing Persistent Objects

- ▶ Commit
 - Save – persist a transient object
 - Update – update an object already in the data store
 - SaveOrUpdate – call appropriate method based on value in id property
 - Only works if `unsaved-value = null` (mapping)
- ▶ Delete – removes an object from the data store
- ▶ Updating Detached Objects
 - Lock associates an object with the current session
 - Ignores changes prior to “Lock” call
 - Should only be used to Initialize proxies

Using Stored Procedures

- ▶ If DBA requires all Stored Procedure access to data store
 - Add tags to the mapping file:
 - `<sql-insert>exec createPerson ?, ?</sql-insert>`
 - `<sql-delete>exec deletePerson ?</sql-delete>`
 - `<sql-update>exec updatePerson ?, ?</sql-update>`
 - Current version has some issues with this
 - Depends on several factors
 - Should be fully functional in 2.5

Custom Connection Provider

- ▶ NHibernate doesn't support encrypted connections
- ▶ Must extend `DriverConnectionProvider`
 - Override `GetNamedConnectionString`

SQL Server Timestamp for Concurrency

- ▶ NHibernate concurrency models:
 - none|version|dirty|all
- ▶ If anything but version, must be in the same session
- ▶ Version is either DateTime or a Sequence
 - Must implement IUserVersionType
 - And hack a little
- ▶ Code Sample

Additional Resources

▶ NHibernate

- <http://tinyurl.com/nhib-doc> – Documentation
- <http://www.manning.com/kuate/> – “NHibernate in Action”
- <http://nhibernate.org> – Project Home
- <http://nhforge.org/Default.aspx> – Community
- <http://groups.google.com/group/nhusers>

Contact Me

- ▶ Phil.japikse@pinnsng.com
- ▶ www.skimedica.com/blog
- ▶ www.twitter.com/skimedica
- ▶ www.pinnsng.com
- ▶ (513) 619-6323