Nhibernate

HARVEY NASH
The Power of Talent

01 July 2013

Tuan Nguyen Manh

Agenda

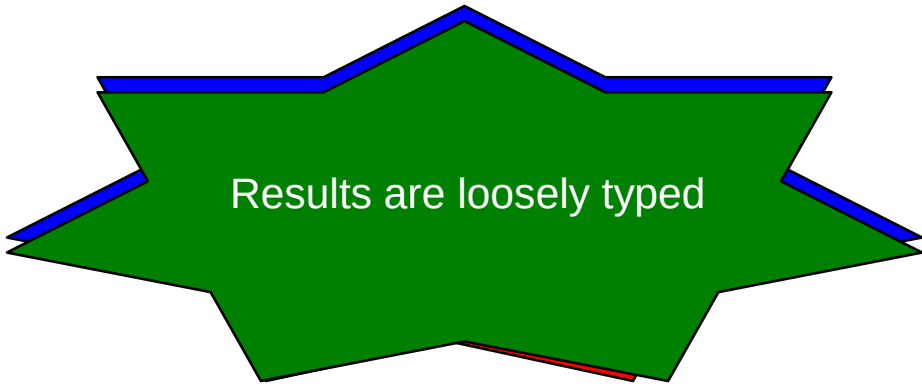
- ORM (Object Relational Mapping) Overview
- Introduction to NHibernate
- Demo
- Exercise
- Q&A

Course Objective

- To deeply understand about Nhibernate framework
- Know how Nhibernate work
- To practice with Nhibernate framework

Here is What You Do…

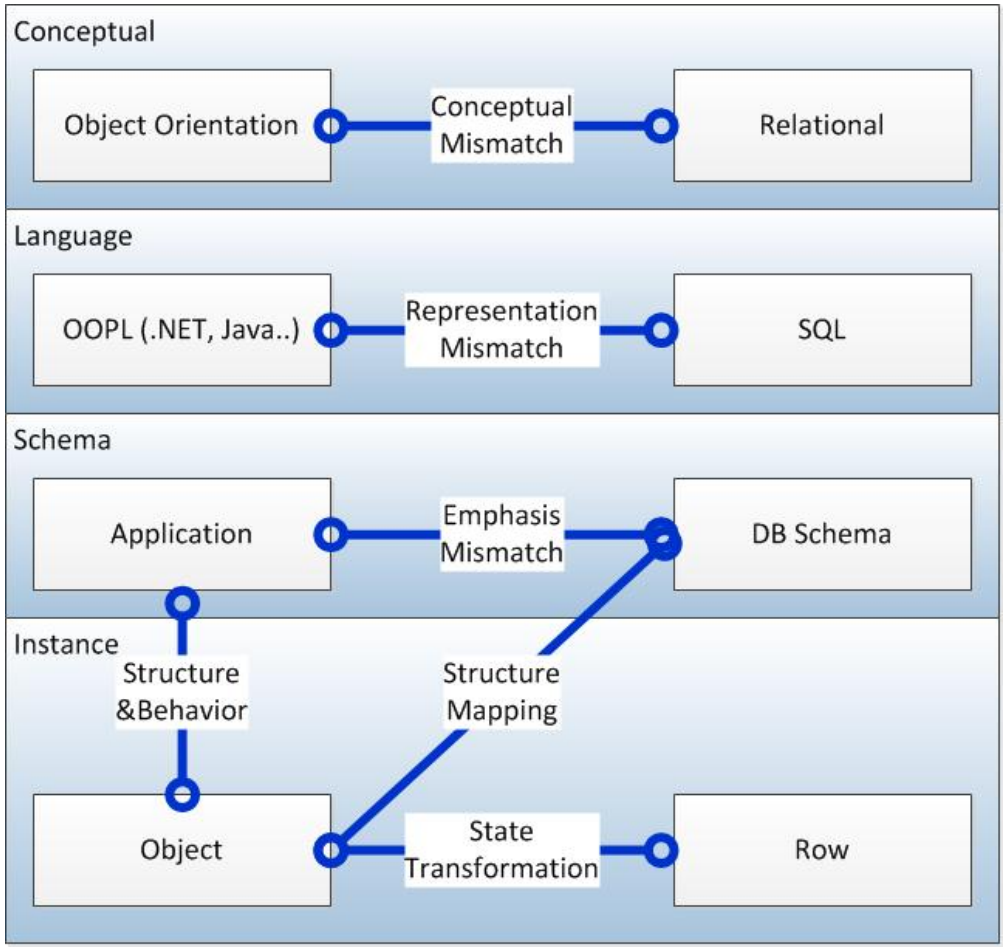
```
using (SQLConnection conn = new SQLConnection(“<conn string>”);
{
    conn.Open();
    SQLCommand cmd = conn.CreateCommand();
    cmd.CommandText = “sp_StoredProc”;
    cmd.parameters.AddWithValue(“@City”, “Dallas”);
    using (SQLDataReader rdr = cmd.ExecuteReader())
    {
       while (rdr.read())
       {
           string name = rdr.GetString(0);
           string city = rdr.GetString(1);
       }
    }
}
```



Powerful, but fragile and time-consuming

Object-oriented Model Vs. Relational Model

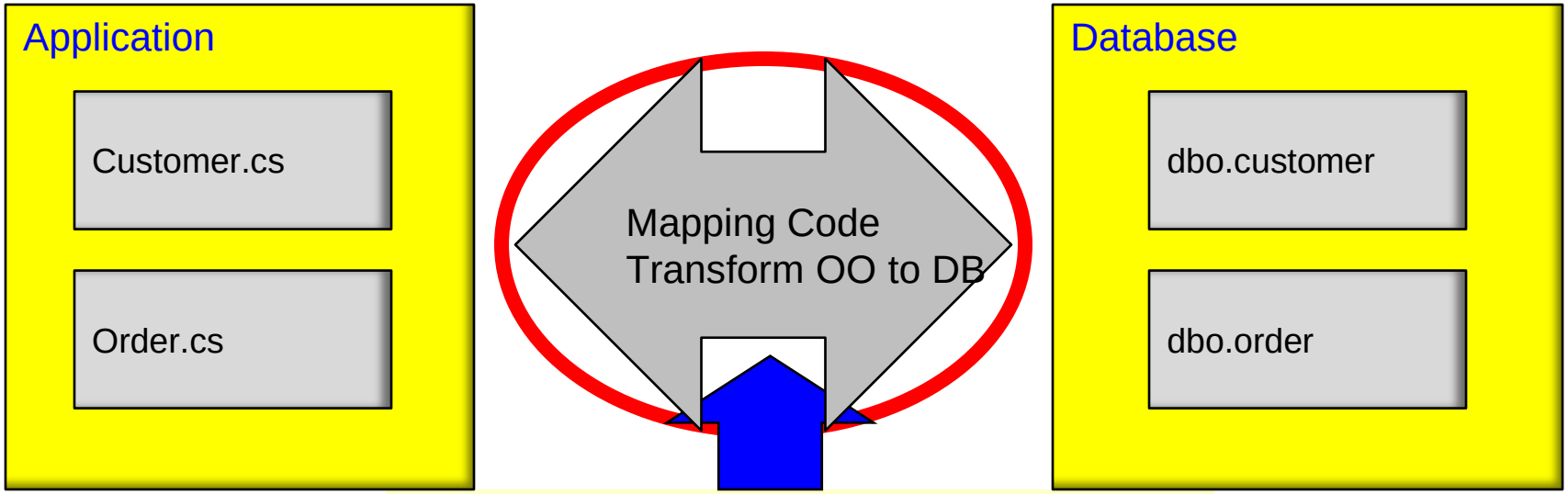
- **Structural**
  - Granularity
  - Subtyping
  - Association
- **Dynamic**
  - Identity
  - Navigation
  - Interface



PARADIGM MISMATCH

ORM (Object Relational Mapping) Overview

- Object Relational Mapping is a **programming technique** for converting data between incompatible type systems in relational databases and object oriented programming languages
  - Objects are hierarchical
  - Databases are relational
  - ORM minimizes Object-relational impedance mismatch



We build plumbing

to move data back and forth
from data store to business objects.

ORM Benefits

Productivity

- Eliminates lots of repetitive code – focus on business logic
- Database schema is generated automatically

Maintainability

- Fewer lines of code – easier to understand
- Easier to manage change in the object model

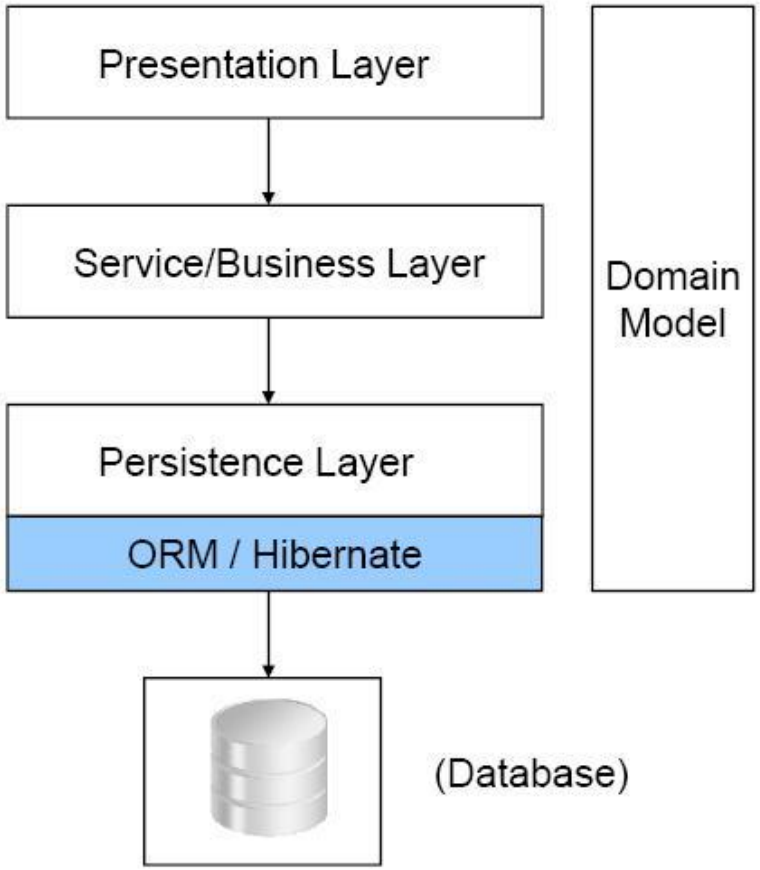
Performance

- Lazy loading – associations are fetched when needed
- Caching

Database vendor independence

- The underlying database is abstracted away
- Can be configured outside the application

ORM into n-tier architecture

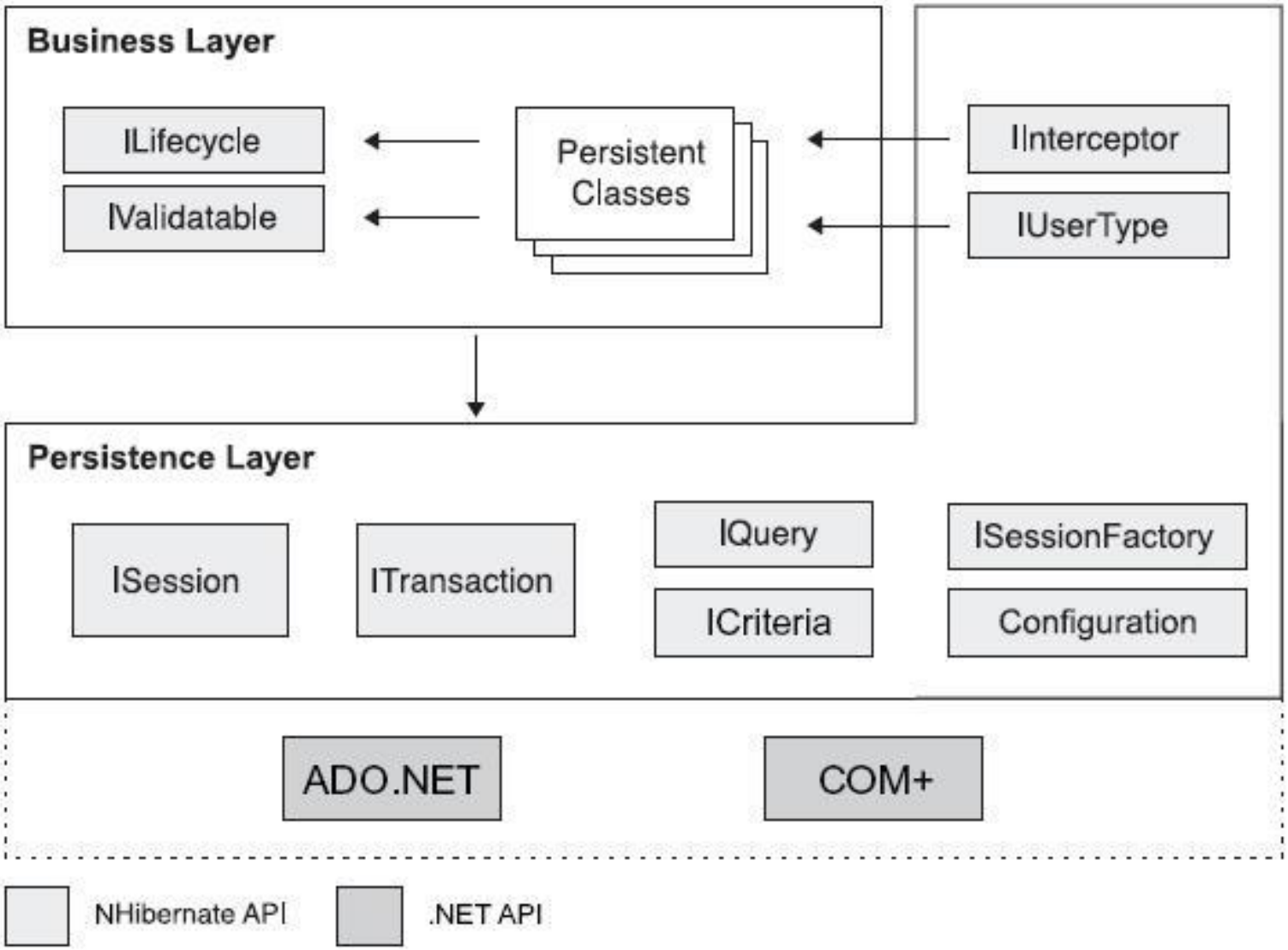


MySQl, Oracle, SQL Server, etc.

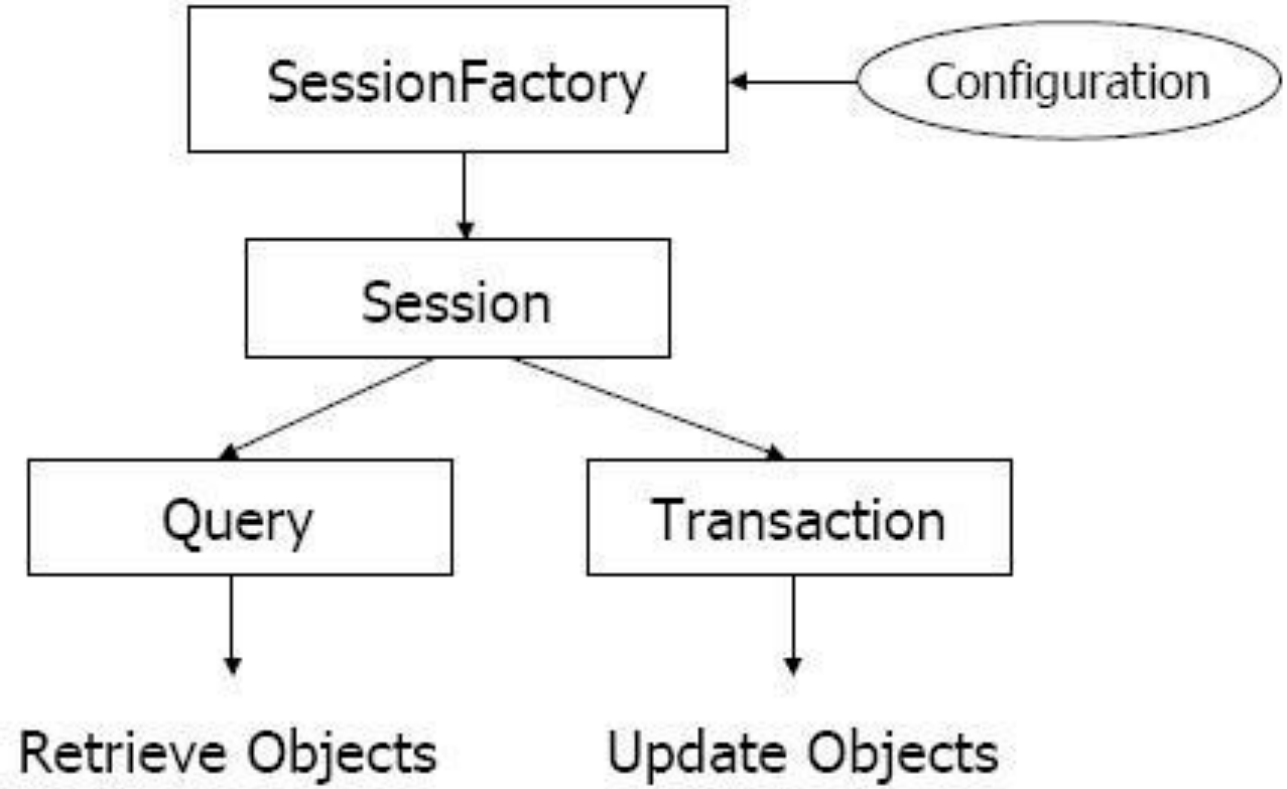
Introduction to NHibernate

- Initially developed for Java
  - created in late 2001 by Gavin King
  - absorbed by the JBossGroup / Red Hat
- Ported to .NET 1.1, 2.0, 3.5
  - Resulting product called “NHibernate”
- All popular databases supported
  - Oracle, SQL Server, DB2, SQLite, PostgreSQL, MySQL, Sybase, Firebird, …
- XML-based configuration files
- Basic CRUD operations
- **Query facilities: HQL, Criteria API, LINQ provider**
- **Multiple levels of caching**
- Good community support
- Free/open source -NHibernateis licensed under the LGPL (Lesser GNU Public License)

High-level overview of the Nhibernate API



Access Persistent Object



Main Interfaces

- ISessionFactory:
  - A threadsafe (immutable) cache of compiled mappings for a single database. A factory for ISession and a client of IConnectionProvider.
- ISession:
  - A single-threaded, short-lived object representing a conversation between the application and the persistent store. Wraps an ADO.NET connection.
- ITransaction:
  - (Optional) A single-threaded, short-lived object used by the application to specify atomic units of work.
- IQuery and ICriteria

Instance States

Transient

- The instance has **not associated with any persistence context**. It has no persistent identity (primary key value).

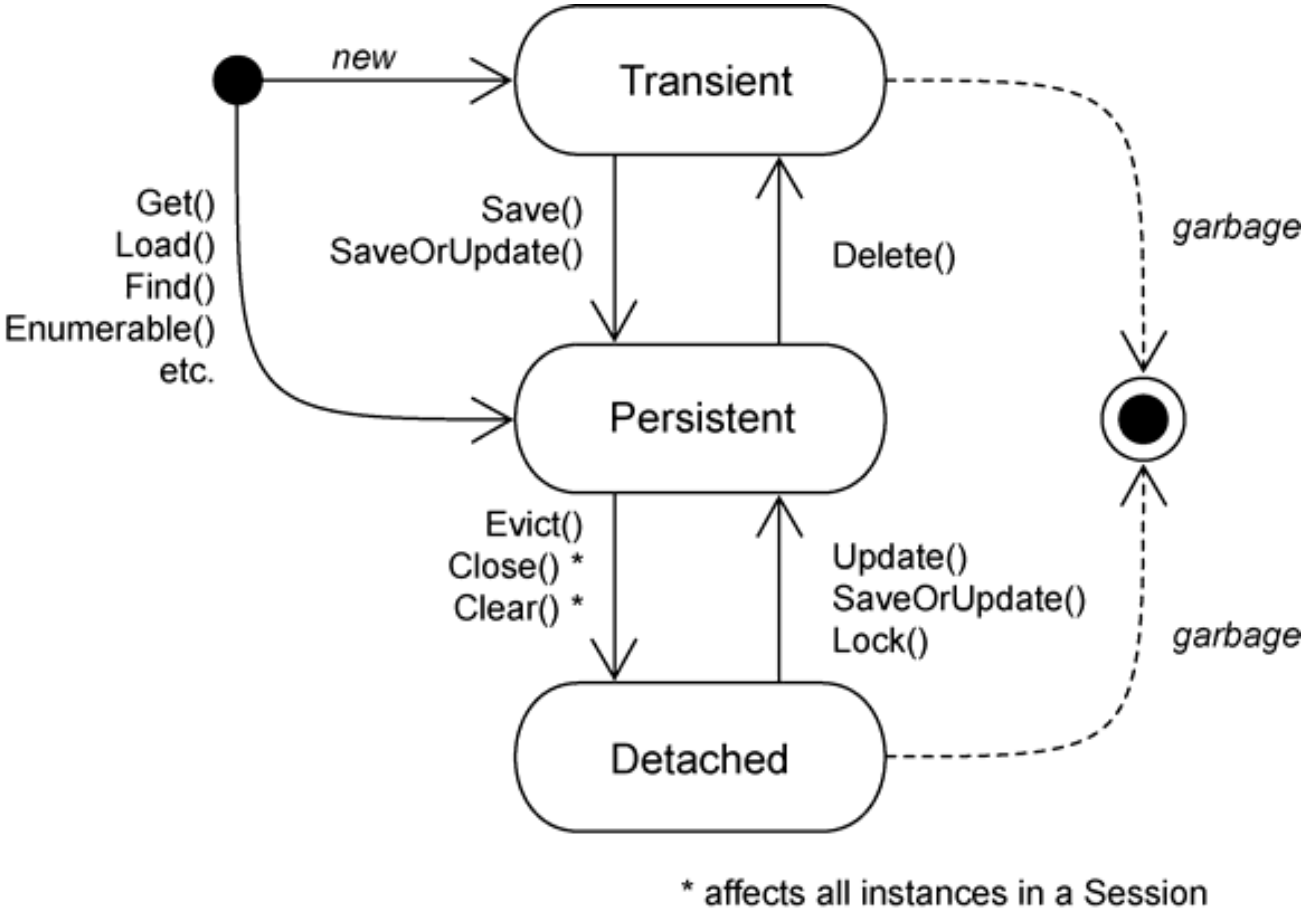
Persistent

- The instance is currently **associated with a persistence context**. It has a persistent identity (primary key value) and, perhaps, a corresponding row in the database.

Detached

- The instance was once **associated with a persistence context, but that context was closed**, or the instance was serialized to another process. It has a persistent identity and, perhaps, a corresponding row in the database.

Instance States

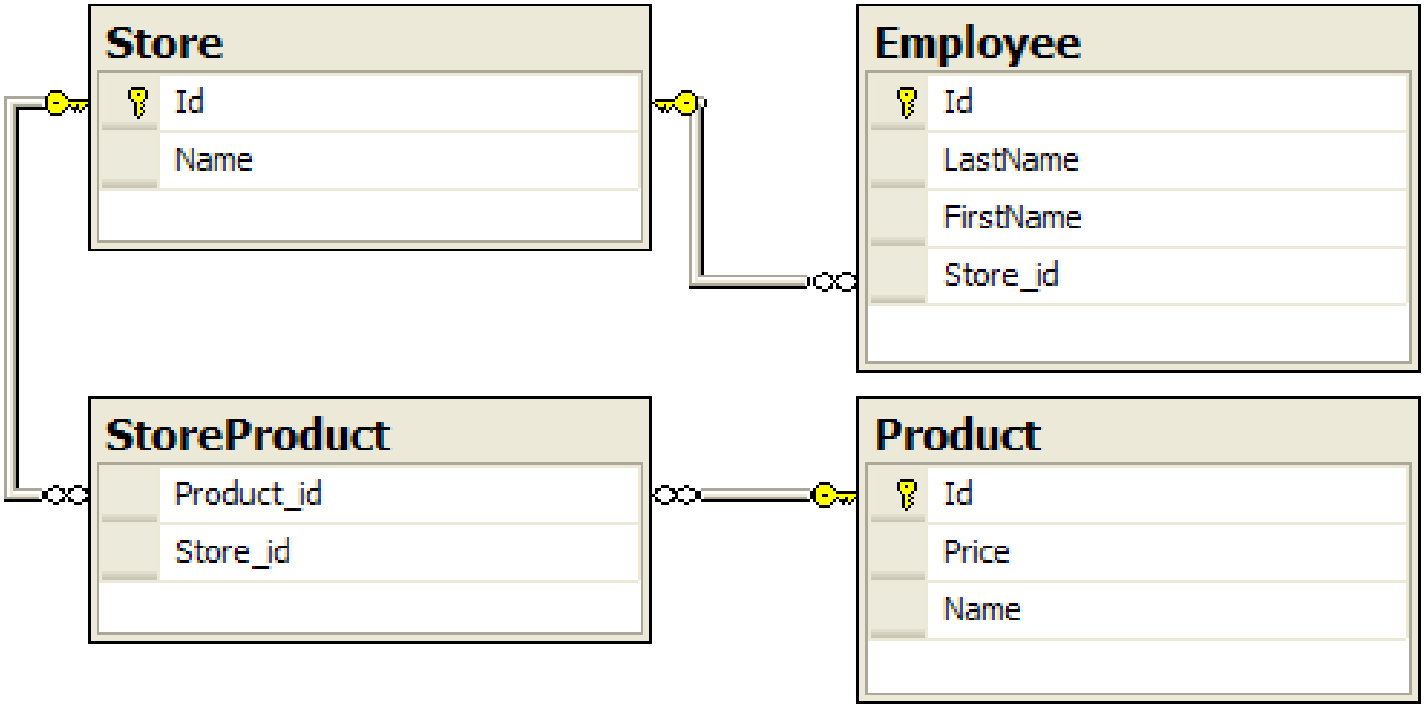


Fluent NHibernate

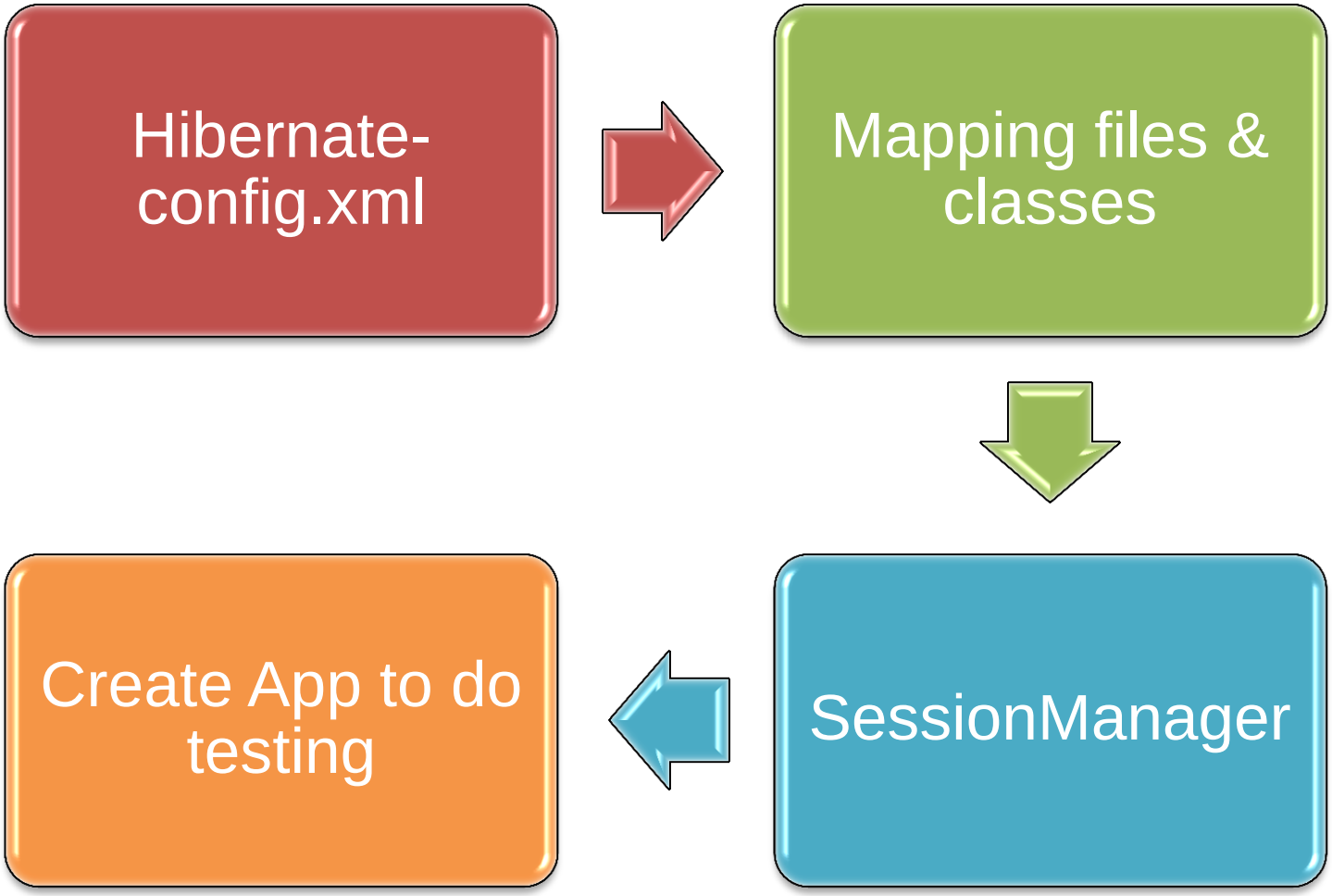
- Started in 2008 based on an idea by Jeremy Miller
- Mapping and configuration in code instead of XML
- Why?
  - XML is verbose
  - Name changes can be caught by the compiler and refactoring tools
  - Eliminates repetition

DEMO

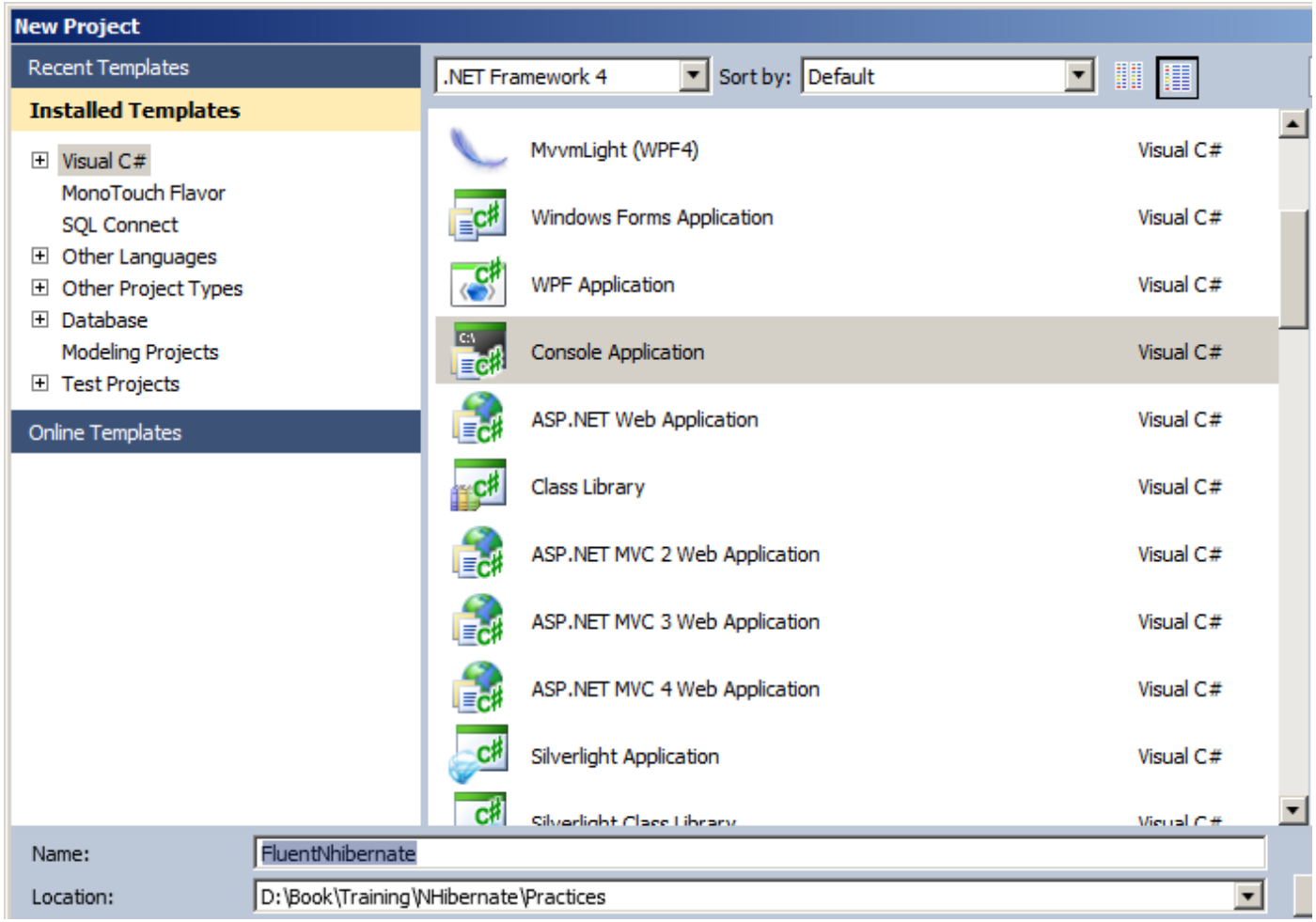
Scenario



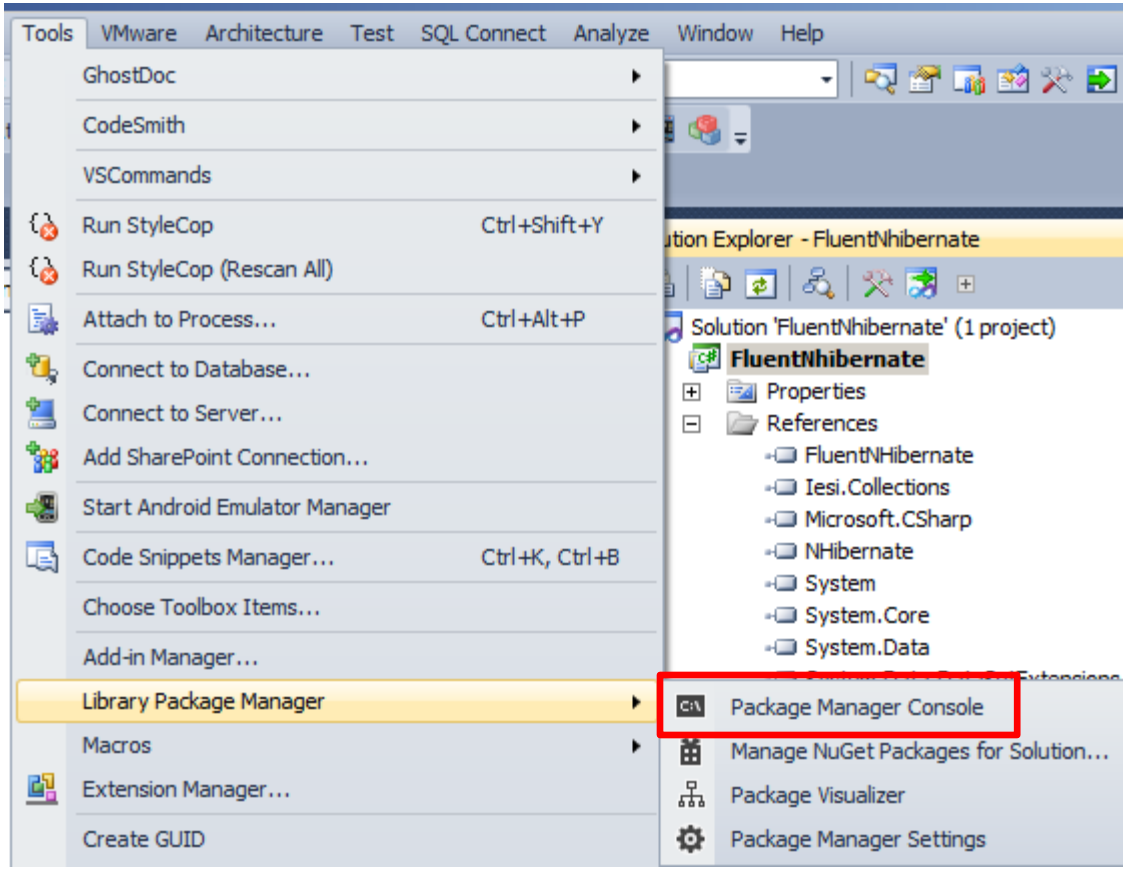
Demo Process



Demo



Demo



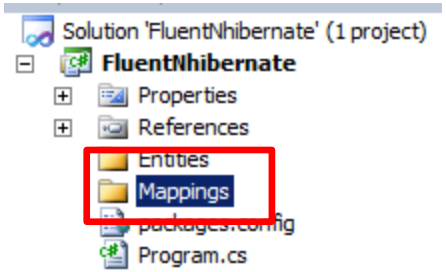
Demo

```
Package Manager Console
Package source: NuGet official package source   Default project: FluentNhibernate1
A new version of NuGet Package Manager is available.   Show Details

Type 'get-help NuGet' to see all available NuGet commands.

PM> Install-Package FluentNHibernate
Attempting to resolve dependency 'NHibernate (≥ 3.3.1.4000)'.
Attempting to resolve dependency 'Iesi.Collections (≥ 3.2 && < 4.0)'.
Successfully installed 'Iesi.Collections 3.2.0.4000'.
Successfully installed 'NHibernate 3.3.3.4001'.
Successfully installed 'FluentNHibernate 1.3.0.733'.
Successfully added 'Iesi.Collections 3.2.0.4000' to FluentNhibernate.
Successfully added 'NHibernate 3.3.3.4001' to FluentNhibernate.
Successfully added 'FluentNHibernate 1.3.0.733' to FluentNhibernate.

PM> Install-Package System.Data.SQLite
Successfully installed 'System.Data.SQLite 1.0.88.0'.
Successfully added 'System.Data.SQLite 1.0.88.0' to FluentNhibernate.
```




```
namespace FluentNhibernate.Entities
{
    public class Employee
    {
        public virtual int Id { get; protected set; }
        public virtual string FirstName { get; set; }
        public virtual string LastName { get; set; }
        public virtual Store Store { get; set; }
    }
}
```

Hibernate-config.xml

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <hibernate-configuration  xmlns="urn:nhibernate-configuration-2.2" >
3    <reflection-optimizer use="true"/>
4    <session-factory>
5      <!-- set connection info. -->
6      <property name="connection.provider">NHibernate.Connection.DriverConnectionProvider</property>
7      <property name="connection.isolation">ReadCommitted</property>
8      <property name="connection.driver_class">NHibernate.Driver.SqlClientDriver</property>
9      <property name="connection.connection_string">
10       Data Source=joeyhur;Initial Catalog=Police;Persist Security Info=True;User ID=sa;Password=sa@123
11     </property>
12
13     <!--assign suitable dialet to different database-->
14     <property name="dialect">NHibernate.Dialect.MsSql2005Dialect</property>
15     <property name="show_sql">true</property>
16
17     <!--use proxy factory -->
18     <property name="proxyfactory.factory_class">NHibernate.ByteCode.Castle.ProxyFactoryFactory, NHibernate.ByteCode.Castle</property>
19
20     <!-- assign mapping files -->
21     <mapping assembly="HelloNHibernate" resource="HelloNHibernate.vo.Contact.hbm.xml"/>
22     <mapping assembly="HelloNHibernate" resource="HelloNHibernate.vo.Employee.hbm.xml"/>
23     <mapping assembly="HelloNHibernate" resource="HelloNHibernate.vo.People.hbm.xml"/>
24   </session-factory>
25 </hibernate-configuration>
```

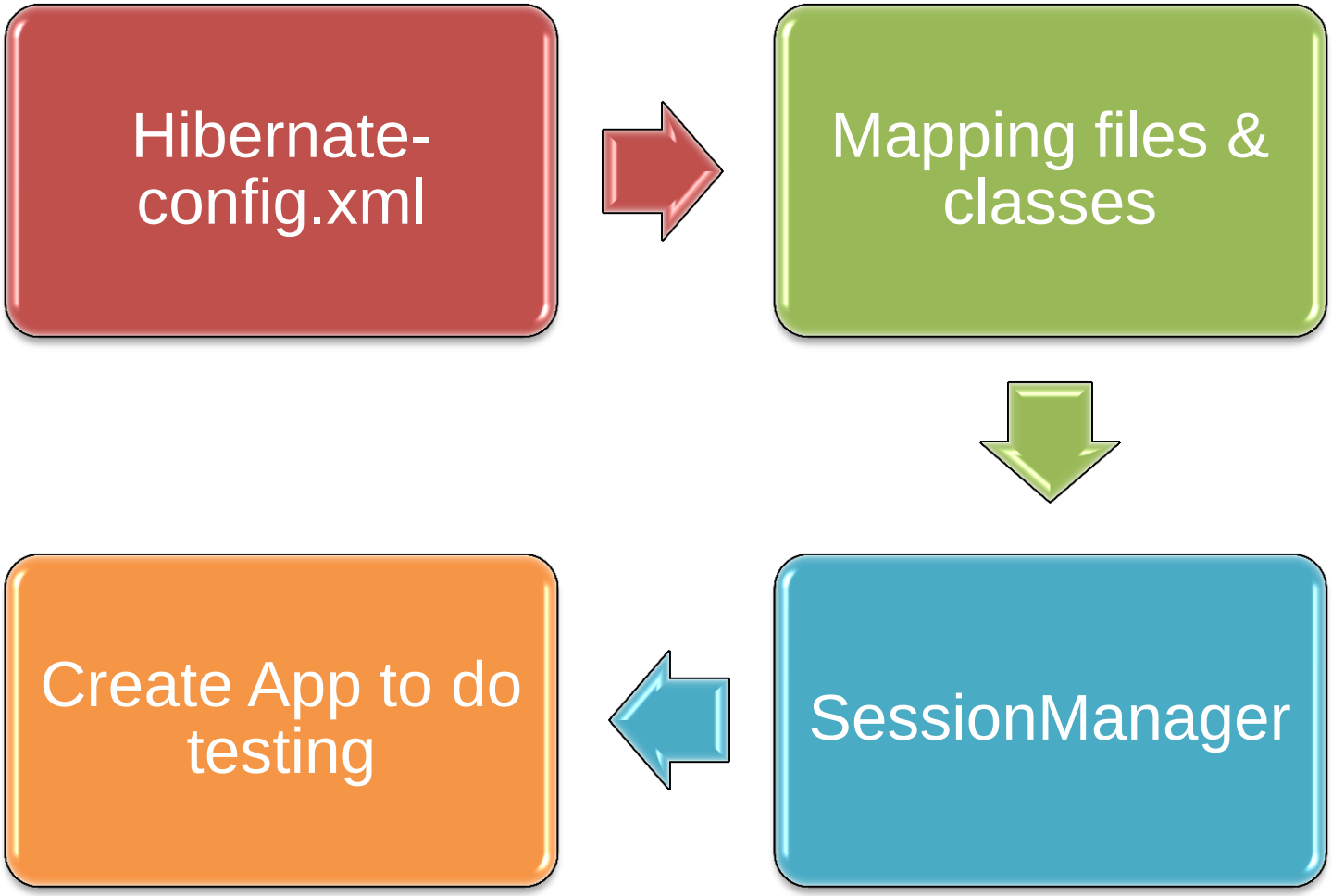
Configuration

- Programmatic Configuration
  - **1- Add hbm.xml files**
    - Configuration cfg = new Configuration()
    - cfg.AddFile(“Student.hbm.xml”);
  - **2- Add entity class (mapping file must be an embedded resource)**
    - Configuration cfg = new Configuration()
    - cfg.AddClass(typeof(Motx.NHDemo.Student));
  - **3- Add an assembly (mapping file must be an embedded resource)**
    - Configuration cfg = new Configuration()
    - cfg.AddAssembly(“Motx.NHDemo.Core”);

Cofiguration via Code in FluentNHibernate

```
private static ISessionFactory CreateSessionFactory()
        {
            return Fluently.Configure()
              .Database(
                SQLiteConfiguration.Standard
                  .UsingFile(DbFile)
              )
              .Mappings(m =>
                m.FluentMappings.AddFromAssemblyOf<Program>())
              .ExposeConfiguration(BuildSchema)
              .BuildSessionFactory();
        }
```

Demo Process



Mapping Concepts

- **ClassMap** declare a persistent class

```
public class PersonMap : ClassMap<Person>
{
  public PersonMap()
  {
  }
}
```

- <id> defines the mapping from that property to the primary key column
- <property> declares a persistent property of the class
  - NHibernate types
  - Dot NET native types
  - Enumeration types
  - Custom types

```
Id(x => x.Id)
  .Column("PersonId")
  .GeneratedBy.Assigned();
```

- <component> maps properties of a child object to columns of the table of a parent class.
- Associations
  - Many-to-One
  - One-to-Many
  - Many-to-Many
  - One-to-One (uncommon)

Mapping Concepts

- **References / many-to-one**

```
public class Book
{
  public Author Author { get; set; }
}
```

```
References(x => x.Author);
```

As with all other fluent mappings, you can chain calls to customise the reference relationship. For example if you wanted to specify the cascade strategy you'd use the `Cascade` property.

```
References(x => x.Author)
  .Column("AuthorId")
  .Cascade.All();
```

Mapping Concepts

- **HasMany / one-to-many**

```
public class Author
{
  public IList<Book> Books { get; set; }
}
```

To map this we use the `HasMany` method, as so:

```
HasMany(x => x.Books);
```

- **HasManyToMany / many-to-many**
  - HasManyToMany works exactly the same as #HasMany / one-to-many, except the underlying database structure it maps to is different.
  - It is use in Many-Many relationship

Mapping Concepts

- **HasOne / one-to-one**

```
HasOne(x => x.Cover);
```

- **Any**
  - The <any> mapping element defines a polymorphic association to classes from multiple tables. This type of mapping always requires more than one column. The first column holds the type of the associated entity. The remaining columns hold the identifier. It is impossible to specify a foreign key constraint for this kind of association, so this is most certainly not meant as the usual way of mapping (polymorphic) associations. You should use this only in very special cases (eg. audit logs, user session data, etc).

```
ReferencesAny(x => x.Author)
   .EntityTypeColumn("Type")
   .EntityIdentifierColumn("Id")
   .IdentityType<int>();
```

Mapping Collections

Element	Description	.NET Type
<set>	An unordered collection that does not allow duplicates.	`Iesi.Collections.ISet` `Iesi.Collections.Generic.ISet<T>`
<list>	An **ordered** collection that allows duplicates	`System.Collections.IList` `System.Collections.Generic.IList<T>`
<bag>	An unordered collection that allow duplicatd	`System.Collections.IList` `System.Collections.Generic.IList<T>`

Mapping Concepts

- **Components**
  - Components are a clever way of mapping a normalized data model into a more reasonable object model.
  - You may have a customer table that has a series of address columns, ideally you'd want the address columns to be mapped into an Address object, rather than just being properties on a Customer; you can do that with components

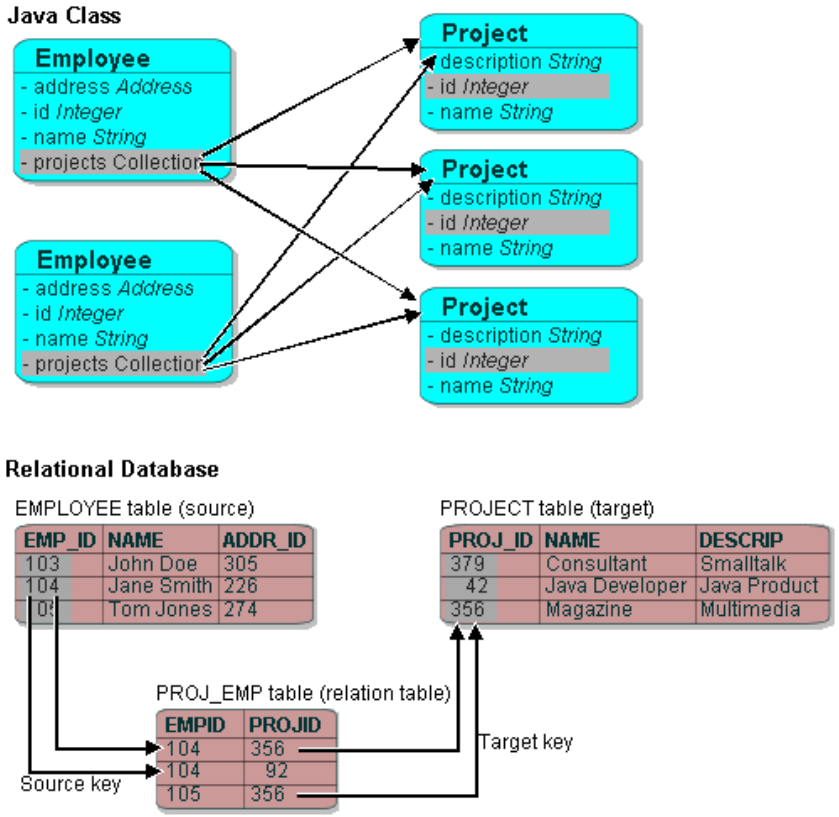
```
Component(x => x.Address, m =>
{
  m.Map(x => x.Number);
  m.Map(x => x.Street);
  m.Map(x => x.PostCode);
});
```

Rules for Mapping

- Create **one table** for each **ordinary class**
  - Make each instance a row in the table
  - Make each simple attribute or qualifier into a column
  - If a **complex attribute** is naturally composed of separate fields, make these into **individual columns**
  - If a complex attribute consumes much space but is **infrequently accessed**, store it in a **separate table**
  - **Make 1-1 or many-1** association into either
    - A column (using a foreign key)
    - A separate table with foreign keys indicating each end of the relation
  - **Decide** on the **key**
    - If unique, one or more of the attributes may constitute the key
    - Otherwise, generate a column that will contain a special object-id as the key

Rules for Mapping (cont.)

- **Many-Many relationship**:
  - Create one table for each association
  - Make the keys of the tables being related to be foreign keys in this new table
  - Can create a special object-id as the key



Demo

```
public class EmployeeMap : ClassMap<Employee>
{
    public EmployeeMap()
    {
        Id(x => x.Id);
        Map(x => x.FirstName);
        Map(x => x.LastName);
        References(x => x.Store);
    }
}
```

```
Id(x => x.Id)
  .Column("PersonId")
  .GeneratedBy.Assigned();
```

```
// nullable

Map(x => x.FirstName) .Nullable();

// not nullable

Map(x => x.FirstName) .Not.Nullable();
```

HARVEY NASH
The Power of Talent

Demo

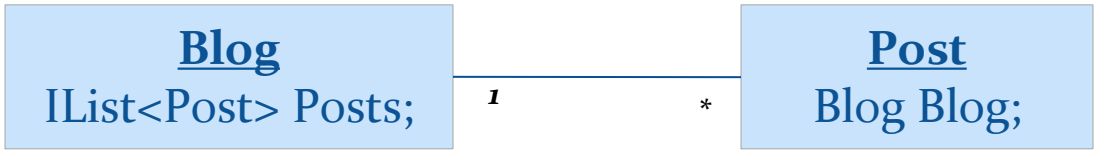
```
public class StoreMap : ClassMap<Store>
    {
        public StoreMap()
        {
            Id(x => x.Id);
            Map(x => x.Name);
            HasMany(x => x.Staff)
              .Inverse()
              .Cascade.All();
            HasManyToMany(x => x.Products)
             .Cascade.All()
             .Table("StoreProduct");
        }
    }
```

Inverse

- **Where do we use inverse?**
  - Inverse is a boolean attribute that can be put on the collection mappings, regardless of collection's role (i.e. within one-to-many, many-to-many etc.), and on join mapping.
  - We can't put inverse on other relation types, like many-to-one or one-to-one.
  - By default, inverse is set to false.
  - Inverse makes little sense for unidirectional relationships, it is to be used only for bidirectional ones.
  - General recommendation is to use inverse="true" on exactly one side of each bidirectional relationship.
  - When we don't set inverse, NHProf will complain about superfluous updates.

Inverse

- NHibernate issue a superfluous update statement



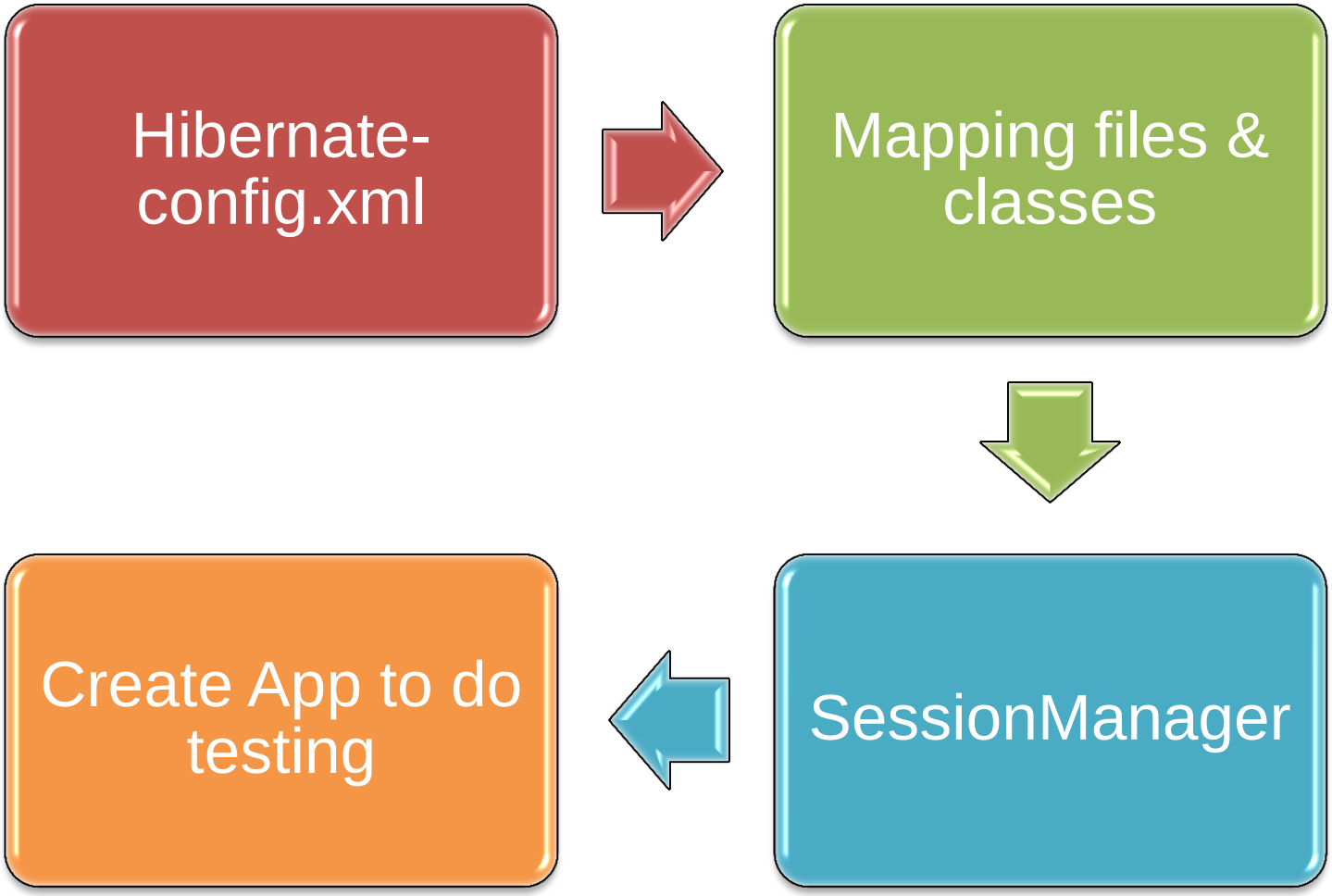

```
INSERT INTO Posts(Title,
                  Text,
                  PostedAt,
                  BlogId,
                  UserId)
VALUES      ('vam' /* @p0 */,
             'abc' /* @p1 */,
             '1/17/2009 5:28:52 PM' /* @p2 */,
             1 /* @p3 */,
             1 /* @p4 */);

select SCOPE_IDENTITY()


UPDATE Posts
SET    BlogId = 1 /* @p0_0 */
WHERE  Id = 22 /* @p1_0 */
```

Left side	Right side	Inverse?
one-to-many	not mapped	makes no sense - left side must be active
one-to-many	many-to-one	right side should be active (left with `inverse="true"`), to save on UPDATEs (unless left side is explicitly ordered)
many-to-many	not mapped	makes no sense - left side must be active
many-to-many	many-to-many	one side should be active (`inverse="false"`), the other should not (`inverse="true"`)

Demo Process



Session Manager

```
8  namespace Web.service
9  {
10     public class SessionManager
11     {
12        private ISessionFactory sessionFactory;
13
14        public SessionManager()
15        {
16           sessionFactory = getSessionFactory();
17        }
18
19        private ISessionFactory getSessionFactory()
20        {
21           Configuration config = new Configuration().Configure();
22           return config.BuildSessionFactory();
23        }
24
25        public ISession getSession()
26        {
27           return sessionFactory.OpenSession();
28        }
29
30     }
31  }
```

[ISession]

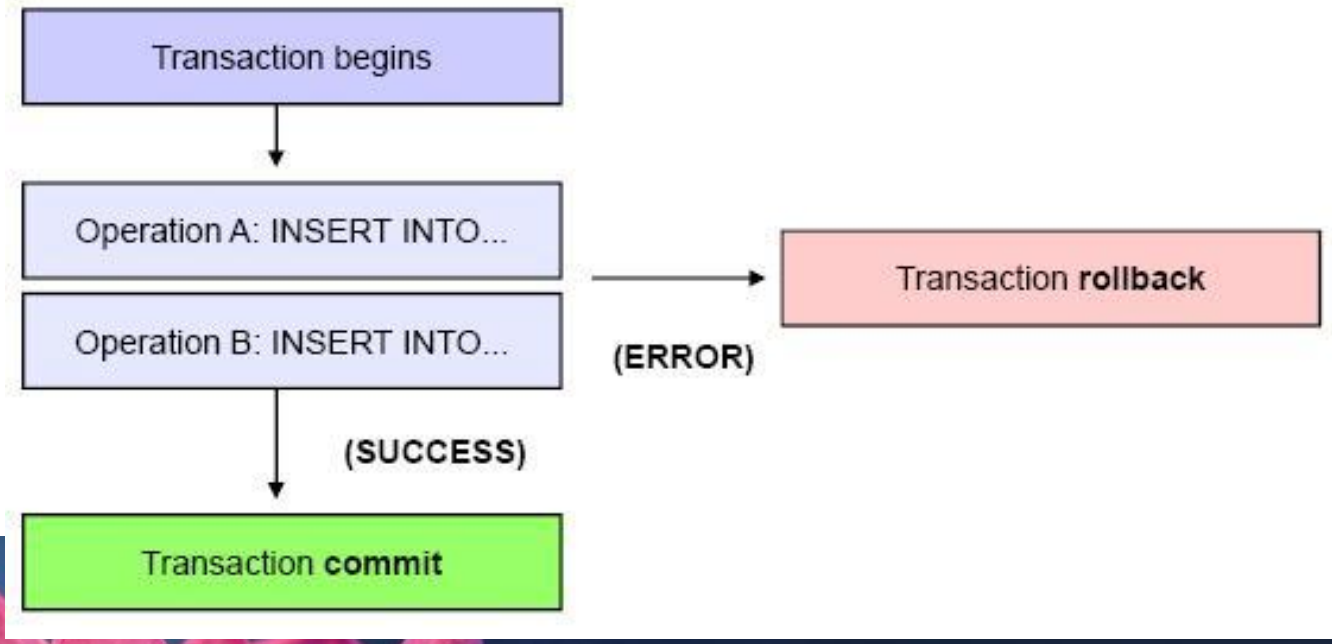
- Obtained from a SessionFactory instance
- Responsible for storing and retrieving objects
- Think of it as a collection of loaded objects related to a single unit of work

FluentNhibernate

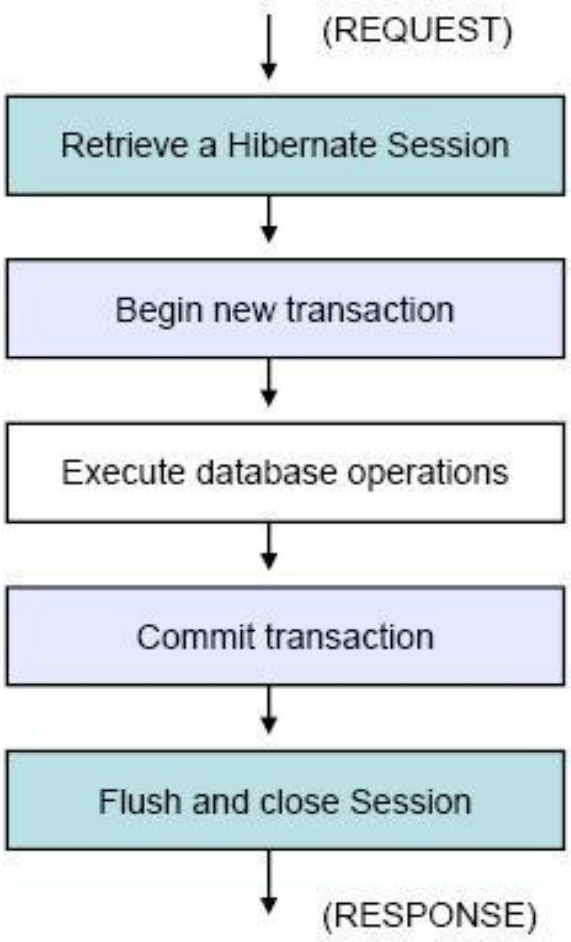
```
private static ISessionFactory CreateSessionFactory()
{
    return Fluently.Configure()
      .Database(
        SQLiteConfiguration.Standard
          .UsingFile(DbFile)
      )
      .Mappings(m =>
        m.FluentMappings.AddFromAssemblyOf<Program>())
      .ExposeConfiguration(BuildSchema)
      .BuildSessionFactory();
}
```

Transactions

- Transaction: A set of database operations which must be executed in entirety or not at all
- Should end either with a commit or a rollback
- All communication with a database has to occur inside a transaction!



Transactions – cont.




```
Session session = sessionFactory.openSession();

Transaction transaction = null;

try
{
   transaction = session.beginTransaction();

   session.save( event );
   session.save( person );

   transaction.commit();
}
catch ( RuntimeException ex )
{
   if ( transaction != null )
   {
      transaction.rollback();
      throw ex;
   }
}
finally
{
   session.close();
}
```

Manipulating Persistent Data

- Saving to the database:

  session.Save(entity);

- Loading one entity from database:

  session.Load(entityType, entityId);

- Loading an entity list from database:

  IQuery query = session.CreateQuery("from Student");
  IList<Student> students = query.List<Student>();

- Updating an entity:

  session.Update(entity);

  session.SaveOrUpdate(entity);

- Deleting an entity:

  session.Delete(entity);

Ending The Session

Flushing the session	• To synchronize the changes with the database • Not needed if the ITransaction API is used
Commit the transaction	• ITransaction.commit() • Flushing will be performed implicitly
Close the session	• The ADO.NET connection will be relinquished by the session
Handle exceptions	• Usually NHibernateException • You should rollback the transaction, close and discard the session instance.

Retrieving Persistent Data – LINQ

- Object oriented querying
- Increase compile-time syntax-checking
- Easy to write
- Queries can be dynamically
- Lambda expressions are awesome

```
var query = from conference in session.Linq<Conference>()
            where conference.StartDate >= date
            orderby conference.StartDate
            select conference;
return query.First();
```

```
session.Query<Product>().Take(2).Where(x => x.Price > 10).ToList();
session.Query<Product>().Where(x => x.Price > 10).Take(2).ToList();
```

Retrieving Persistent Data – Criteria API

- Object oriented querying
- Increase compile-time syntax-checking
- Easy to write
- Hard to read

```
ICriteria crit = sess.CreateCriteria(typeof(Cat));
crit.SetMaxResults(50);
List topCats = crit.List();


IList cats = sess.CreateCriteria(typeof(Cat))
 .Add( Restrictions.Like("Name", "Fritz%"))
 .Add( Restrictions.Between("Weight", minWeight,
      maxWeight))
 .List();
```

Retrieving Persistent Data – HQL

- String based querying
- Object-Oriented SQL
- Similar to SQL
- Speak in terms of objects
- Case sensitive
- Very flexible
- Zero compile-time syntax-checking

```
• from Customer c where c.Name like :name

• select count(*) from Customer c
```

Retrieving Persistent Data – Query by Example

- Powerful way to (simply) return a group of like objects from the DB.Wonderfully simple to work with
  - Great way to quickly process a "…where A=<something> and B=<something> and C=<something>…"

```
Cat cat = new Cat();
cat.Sex = 'F';
cat.Color = Color.Black;
List results = session.CreateCriteria(typeof(Cat))
      .Add( Example.Create(cat) )
      .List();
```

Retrieving Persistent Data – Native SQL

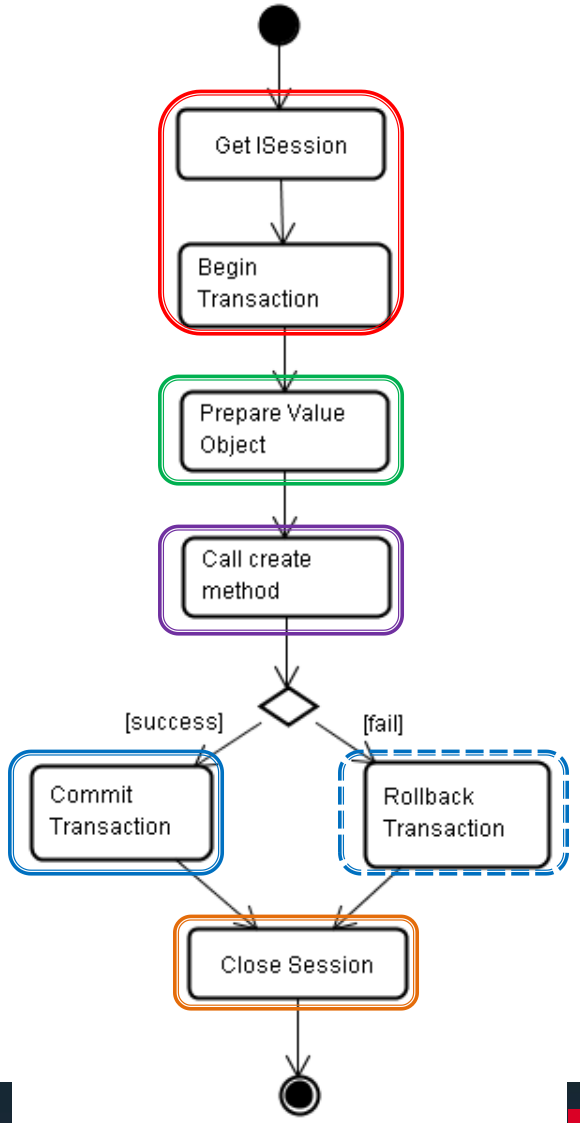
- You can submit SQL statements to Nhibernate if the other methods of querying a database do not fit your needs utilize database specific features

```
• sess.CreateSQLQuery("SELECT * FROM CATS")
      .AddScalar("ID", NHibernateUtil.Int32)
      .AddScalar("NAME", NHibernateUtil.String)
      .AddScalar("BIRTHDATE", NHibernateUtil.Date);


• sess.CreateSQLQuery("SELECT * FROM CATS")
            .AddEntity(typeof(Cat));
```

Example - Create App to do testing

```
41      protected void save(object sender, EventArgs e) {
42          ITransaction tx = null;
43          ISession session = null;
44          People result = null;
45          try
46          {
47              session = new Web.service.SessionManager().getSession();
48              tx = session.BeginTransaction();
49
50              People vo = new People();
51              vo.Name = NameTextBox.Text;
52              vo.Gender = GenderList.SelectedValue;
53
54              result = new PeopleDao().create(session, vo);
55
56              tx.Commit();
57          }
58          catch (Exception ex)
59          {
60              if (tx != null) tx.Rollback();
61              logger.Error(ex);
62              throw ex;
63          }
64          finally {
65              if (session != null) session.Close();
66              Response.Redirect("UpdatePeople.aspx?Id="+result.Id);
67          }
68      }
```



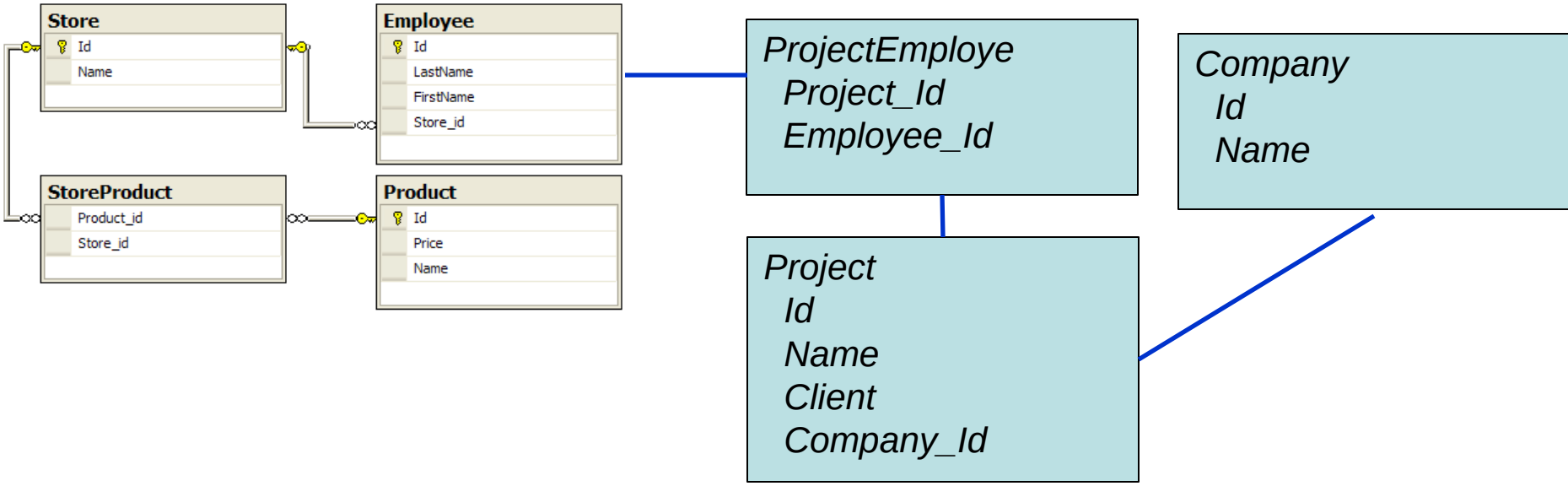
Exercise

- Fluent NHibernate in a Nutshell: https://github.com/jagregory/fluent-nhibernate/wiki/Getting-started
  - List all Products which have Store name like ‘HVN’ , order by Category via Linq and Native SQL
  - List all Employee which have Produce price > 3000

- Note: if can’t download from NuGet, please access link to get all Libs: \\192.168.198.143\Libs

Exercise

- Add more Project and Company as following relationship
- Insert data for Project and Company tables



References

- http://ormeter.net/
- http://channel9.msdn.com/Events/TechEd/Australia/2010/ARC303
- https://services.brics.dk/java/courseadmin/PaSOOS/documents/getDocument/tooltech-orm.pdf?d=62667
- http://www.ceciis.foi.hr/app/index.php/ceciis/2009/paper/download/231/221


Q&A