ЛЕКЦІЯ № 9

NHibernate

Посилання та література

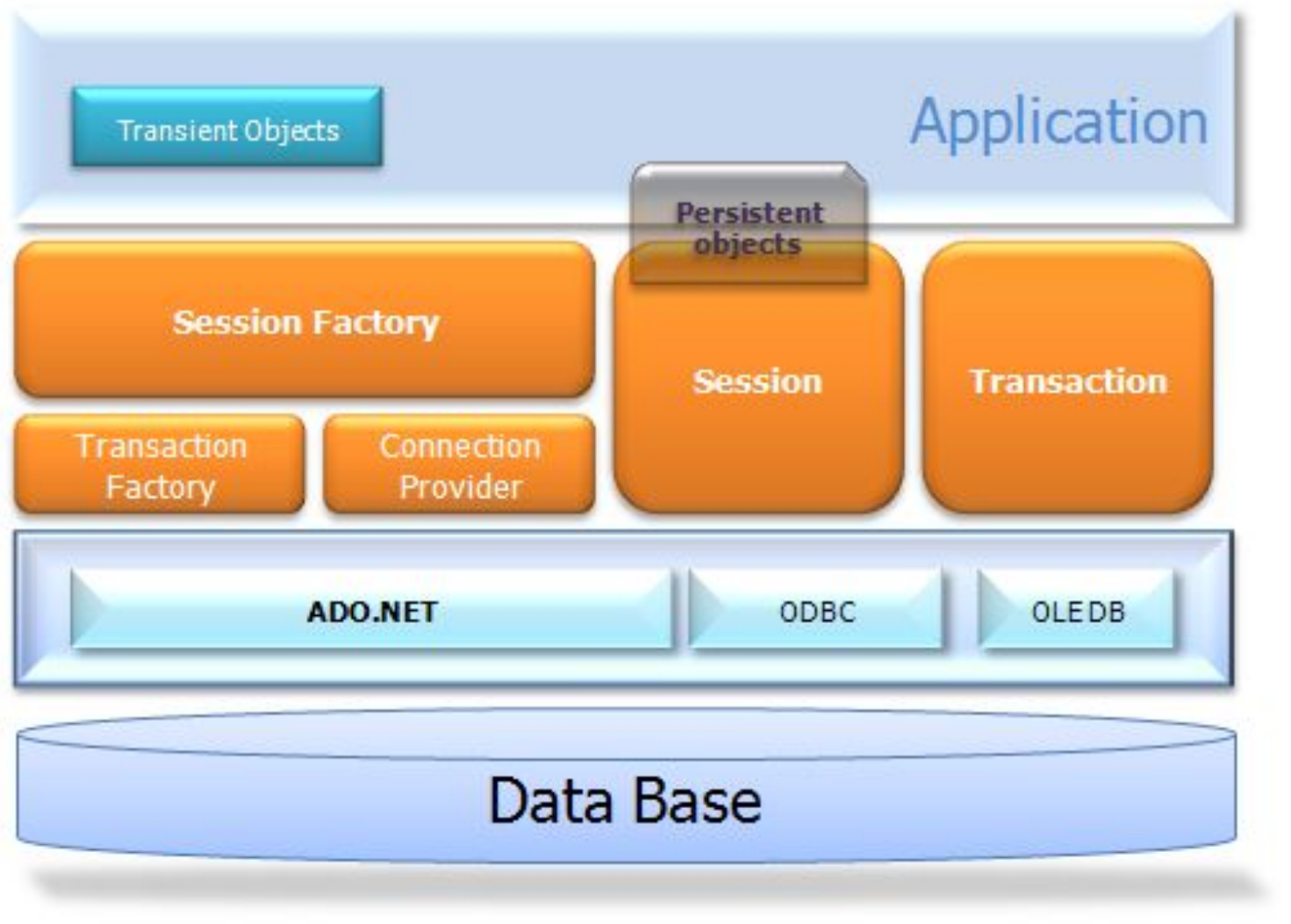
Де скачати NHibernate?

http://sourceforge.net/projects/nhibernate/

Література:

1. Pierre Henri Kuaté, Tobin Harris, Christian Bauer, Gavin King. NHibernate in Action;
2. NHibernate Reference Documentation.

Архітектура NHibernate



Створення класу предметної області

```
namespace lab4.Domain{
    class Cat{
        public virtual long Id { get; set; }

        public virtual string Name { get; set; }

        public virtual int Age { get; set; }

        public virtual float Weight { get; set; }
    }
}
```

Файл мапінгу класу предметної області

```
<?xml version="1.0" encoding="UTF-8"?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
        namespace="lab4.Domain" assembly="lab4">
    <class name="Cat" table="Cats">
        <!– Оголошення первинного ключа та його політики генерування-->
        <id name="Id">
            <column name="CatID" sql-type="bigint" not-null="true"/>
            <generator class="sequence"/>
        </id>
        <!--Оголошення полів об’єкта Cat-->
        <property name="Name"/>
        <property name="Age"/>
        <property name="Weight"/>
    </class>
</hibernate-mapping>
```

Конфігурація NHibernate

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-configuration  xmlns="urn:nhibernate-configuration-2.2" >
    <session-factory>
        <property name="connection.provider">
            NHibernate.Connection.DriverConnectionProvider
        </property>
        <property name="dialect">
            NHibernate.Dialect.PostgreSQLDialect
        </property>
        <property name="connection.driver_class">
            NHibernate.Driver.NpgsqlDriver
        </property>
        <property name="connection.connection_string">
            Server=127.0.0.1;Port=5432;User Id=postgres;Password=1111;Database=animals;
        </property>
        <property name="proxyfactory.factory_class">
            NHibernate.ByteCode.Spring.ProxyFactoryFactory, NHibernate.ByteCode.Spring
        </property>
        <!--<property name="hbm2ddl.auto">create</property>-->
        <mapping resource="lab4.Mappings.Cat.hbm.xml" assembly="lab4" />
    </session-factory>
</hibernate-configuration>
```

Відкриття сесії

```
static ISession openSession(){
      ISessionFactory factory = null;
          ISession session = null;
          if (factory == null){
              Configuration configuration = new Configuration();
              configuration.Configure();
              factory = configuration.BuildSessionFactory();
              session = factory.OpenSession();
          }
          return session;
}
```

Робота з сесією

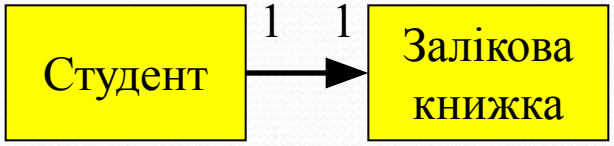
```
static void Main(string[] args){
    ISession session = openSession();

    ITransaction transaction = session.BeginTransaction();
    session.SaveOrUpdate("Cats", new Cat {Name = "Борис"});
    transaction.Commit();

    IList<Cat> cats = session.CreateSQLQuery("SELECT * FROM cats")
       .AddEntity("Cat", typeof(Cat))
            .List<Cat>();

    foreach (var cat in cats){
      Console.WriteLine("{0}\t{1}", cat.Id, cat.Name);
    }
    Console.ReadLine();
}
```

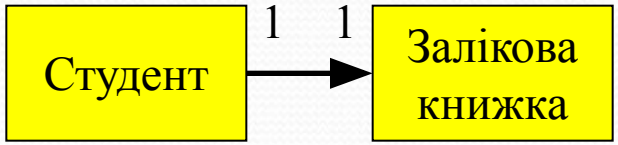
Зв’язок One-to-One (класи)




```
class Student{
      public virtual long StudentID { get; set; }
      public virtual string Name { get; set; }
      public virtual string Surname { get; set; }
      public virtual boolean Contractor { get; set; }
      public virtual RecordBook recordBook { get; set; }
   }

 class RecordBook{
        public virtual long RecordBookID { get; set; }
        public virtual int Number { get; set; }
 }
```

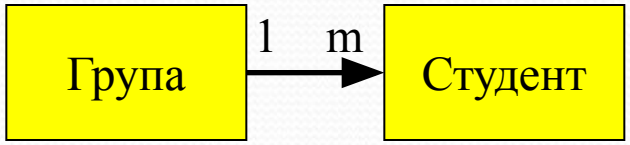
Зв’язок One-to-One (файли мапінгів)




```
<class name="Student" table="Students">
    <id name="StudentID">
        <column name="StudentID" sql-type="bigint" not-null="true"/>
        <generator class="native"/>
    </id>
    <property name="Name"/>
    <property name="Surname"/>
    <property name="Contractor"/>
    <one-to-one name="recordBook"
      class="RecordBook, NHibernateOneToOne"/>
</class>
```

```
<class name="RecordBook" table="RecordBooks">
    <id name="RecordBookID">
        <column name="RecordBookID " sql-type="bigint" not-null="true"/>
        <generator class="native"/>
    </id>
    <property name="Number"/>
</class>
```

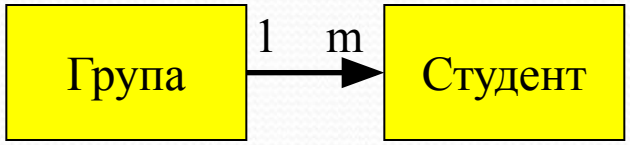
Зв’язок One-to-Many (IList)




```
class Group{
      public virtual long GroupID { get; set; }
      public virtual string GroupName { get; set; }
      public virtual string CuratorName { get; set; }
      public virtual string HeadmanName { get; set; }
      public virtual IList<Student> StudentList { get; set; }
}
```

```
<list name="StudentList" cascade="all">
     <key column="GroupID"/>
     <index column="Index"/>
     <one-to-many class="Student, NHibernateOneToMany"/>
</list>
```

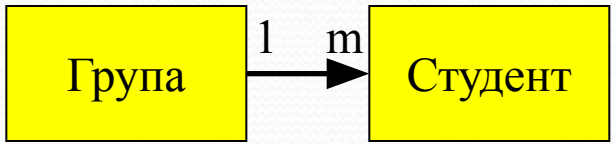
Зв’язок One-to-Many (ISet)




```
class Group{
        public virtual long GroupID { get; set; }
        public virtual string GroupName { get; set; }
        public virtual string CuratorName { get; set; }
        public virtual string HeadmanName { get; set; }
        public virtual ISet<Student> StudentSet { get; set; }
}
```

```
<set name="StudentSet" cascade="all">
    <key column="GroupID"/>
    <one-to-many class="Student, NHibernateOneToMany"/>
</set>
```

Зв’язок One-to-Many (IDictionary)




```
class Group{
        public virtual long Id { get; set; }
        public virtual string Name { get; set; }
        public virtual string Curator { get; set; }
        public virtual string Headman { get; set; }
        public virtual IDictionary<Student, string> StudentDictionary { get; set; }
}
```

```
<map name="StudentDictionary" cascade="all">
     <key column="GroupID"/>
     <index column="Name" type="String"/>
     <one-to-many class="Student, NHibernateOneToMany"/>
</map>
```

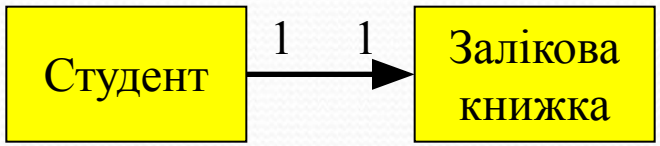
Fluent NHibernate

Fluent Nhibernate – бібліотека, яка дозволяє описувати правила проектування об’єктів на реляційну базу даних з використанням класів, а не XML-файлів, як це було у бібліотеці NHibernate.

Домашня адреса:

http://fluentnhibernate.org/

Зв’язок One-to-One (класи)




```
public abstract class EntityBase{
        public virtual long ID { get; set; }
}
```

```
public class Student : EntityBase{
        public virtual string Name { get; set; }
        public virtual string Surname { get; set; }
        public virtual boolean Contractor { get; set; }
        public virtual RecordBook RecordBook { get; set; }
}
```

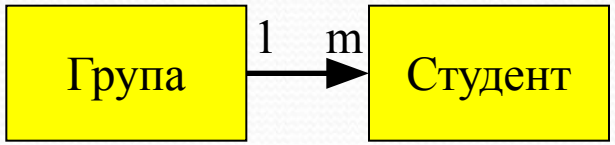
```
public class RecordBook : EntityBase{
        public virtual int Number { get; set; }
        public virtual Student Student { get; set; }
}
```

Зв’язок One-to-One (класи мапінгу)

```
public class StudentMap : ClassMap<Student>{
        public StudentMap(){
            Table("Students");
            Id(x => x.ID).GeneratedBy.Native();
            Map(x => x.Name);
            Map(x => x.Surname);
            Map(x => x.Contractor);
            HasOne(x => x.RecordBook).ForeignKey("StudentID").Cascade.All();
        }
}
```

```
public RecordBookMap(){
            Table("RecordBooks");
            Id(x => x.ID).GeneratedBy.Native();
            Map(x => x.Number);
            References(x => x.Student).Column("StudentID").Cascade.All();
}
```

Зв’язок One-to-Many (класи)




```
public abstract class EntityBase{
        public virtual long ID { get; set; }
}
```

```
public class Group : EntityBase{
        private IList<Student> studentList = new List<Student>();
        public virtual string GroupName { get; set; }
        public virtual string CuratorName { get; set; }
        public virtual string HeadmanName { get; set; }
        public virtual IList<Student> StudentList{
            get { return studentList; }  set { studentList = value; }
        }
}
```

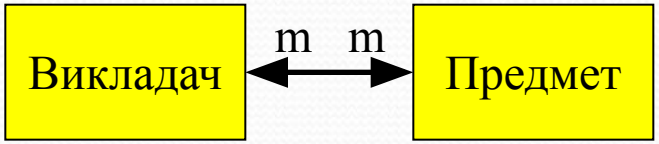
```
public class Student : EntityBase{
        public virtual string Name { get; set; }
        public virtual string Surname { get; set; }
        public virtual char Contractor { get; set; }
        public virtual Group Group { get; set; }
}
```

Зв’язок One-to-Many (класи мапінгу)

```
public class GroupMap : ClassMap<Group>{
        public GroupMap(){
            Table("Groups");
            Id(x => x.ID).GeneratedBy.Native();
            Map(x => x.GroupName);
            HasMany(x => x.StudentList)
        .KeyColumns.Add("GroupID")
        .Inverse().Cascade.All();
        }
}
```

```
public class StudentMap : ClassMap<Student>{
        public StudentMap(){
            Table("Students");
            Id(x => x.ID).GeneratedBy.Native();
            Map(x => x.Surname);
            References(x => x.Group).Column("GroupID").Cascade.All();
        }
    }
```

Зв’язок Many-to-Many (класи)




```
public abstract class EntityBase{
        public virtual long ID { get; set; }
}
```

```
public class Subject : EntityBase{
        private IList<Teacher> teacherList = new List<Teacher>();
        public virtual string SubjectName { get; set; }
        public virtual int HoursNumber { get; set; }
        public virtual IList<Teacher> TeacherList{
            get { return teacherList; } set { teacherList = value; }
        }
}
```

```
public class Teacher : EntityBase{
        private IList<Subject> subjectList = new List<Subject>();
        public virtual string Name { get; set; }
        public virtual string Surname { get; set; }
        public virtual IList<Subject> SubjectList{
            get { return subjectList; } set { subjectList = value; }
        }
}
```

Зв’язок Many-to-Many (класи мапінгу)

```
public class SubjectMap : ClassMap<Subject>{
    public SubjectMap(){
        Table("Subjects");
        Id(x => x.ID).GeneratedBy.Native();
        Map(x => x.SubjectName);
        Map(x => x.HoursNumber);
        HasManyToMany(x => x.TeacherList).Table("TeacherSubject")
            .ParentKeyColumn("SubjectID").ChildKeyColumn("TeacherID");
    }
}
```

```
public class TeacherMap : ClassMap<Teacher>{
    public TeacherMap(){
        Table("Teachers");
        Id(x => x.ID).GeneratedBy.Native();
        Map(x => x.Name);
        Map(x => x.Surname);
        HasManyToMany(x => x.SubjectList).Table("TeacherSubject")
            .ParentKeyColumn("TeacherID").ChildKeyColumn("SubjectID");
    }
}
```

Створення сесії

```
private static ISession openSession(){
        ISessionFactory factory = null;
        ISession session = null;
        Assembly mappingsAssemly = Assembly.GetExecutingAssembly();
        if (factory == null){
            factory = Fluently.Configure()
                .Database(PostgreSQLConfiguration.PostgreSQL82.ConnectionString(c => c
                    .Host("localhost")
                    .Port(5432)
                    .Database("shop")
                    .Username("postgres")
                    .Password("1111")))
                .Mappings(m => m.FluentMappings.AddFromAssembly(mappingsAssemly))
                .ExposeConfiguration(BuildSchema)
                .BuildSessionFactory();
            session = factory.OpenSession();
        }
        return session;
}
```

```
private static void BuildSchema(Configuration config){
        new SchemaExport(config).Create(false, true);
}
```

Запити у NHibernate

- Запити Native SQL;
- Запити по критерію (Criteria queries);
- Запити по прикладу (Example queries);
- Запити з використанням мови запитів HQL.

Запити Native SQL

```
private static IList<Student> getStudentsByGroup(ISession session, string groupName){
        IList<Student> list = session.CreateSQLQuery(
                "SELECT Students.* FROM Students JOIN Groups "+
                "ON Students.GroupID = Groups.ID "+
                "WHERE Groups.GroupName='" + groupName + "'")
                .AddEntity("Student", typeof(Student))
                .List<Student>();
            return list;
}
```

```
IList<Student> list = getStudentsByGroup(session, "KI-141");
foreach (Student student in list){
    Console.WriteLine("{0} {1}\t\t{2}", student.LastName,
                student.FirstName, student.Contractor);
}
Console.ReadLine();
```

Запити по критерію

```
ICriteria criteria1 = session.CreateCriteria(typeof(Student))
        .Add(Expression.Eq("Contractor", false))
        .SetMaxResults(5);
List<Student> list = criteria1.List<Student>();
```

```
ICriteria criteria2 = session.CreateCriteria(typeof(Student))
        .Add(Expression.Like("Name", "Олекс%"));
List<Student> list = criteria2.List<Student>();
```

```
ICriteria criteria3 = session.CreateCriteria(typeof(Student))
            .Add(Expression.Between("BirthYear", 1995, 1996));
List<Student> list = criteria3.List<Student>();
```

```
ICriteria criteria4 = session.CreateCriteria(typeof(Student))
        .Add(Expression.Like("Name", "Олекс%"))
        .AddOrder(Order.Desc("BirthYear"))
        .AddOrder(Order.Asc("Surname"));
List<Student> list = criteria4.List<Student>();
```

Запити по прикладу

```
Student s = new Student();
s.Contractor = false;
s.BirthYear = 1998;
ICriteria criteria4 = session.CreateCriteria(typeof(Student))
        .Add(Example.Create(s));
list = criteria4.List<Student>();
```

```
Student s1 = new Student();
s.Contractor = false;
s.BirthYear = 1996;
Example example = Example.Create(s1)
            .ExcludeZeroes()
            .ExcludeProperty("Year")
            .IgnoreCase()
            .EnableLike();
ICriteria criteria5 = session.CreateCriteria(typeof(Student))
            .Add(example);
list = criteria5.List<Student>();
```

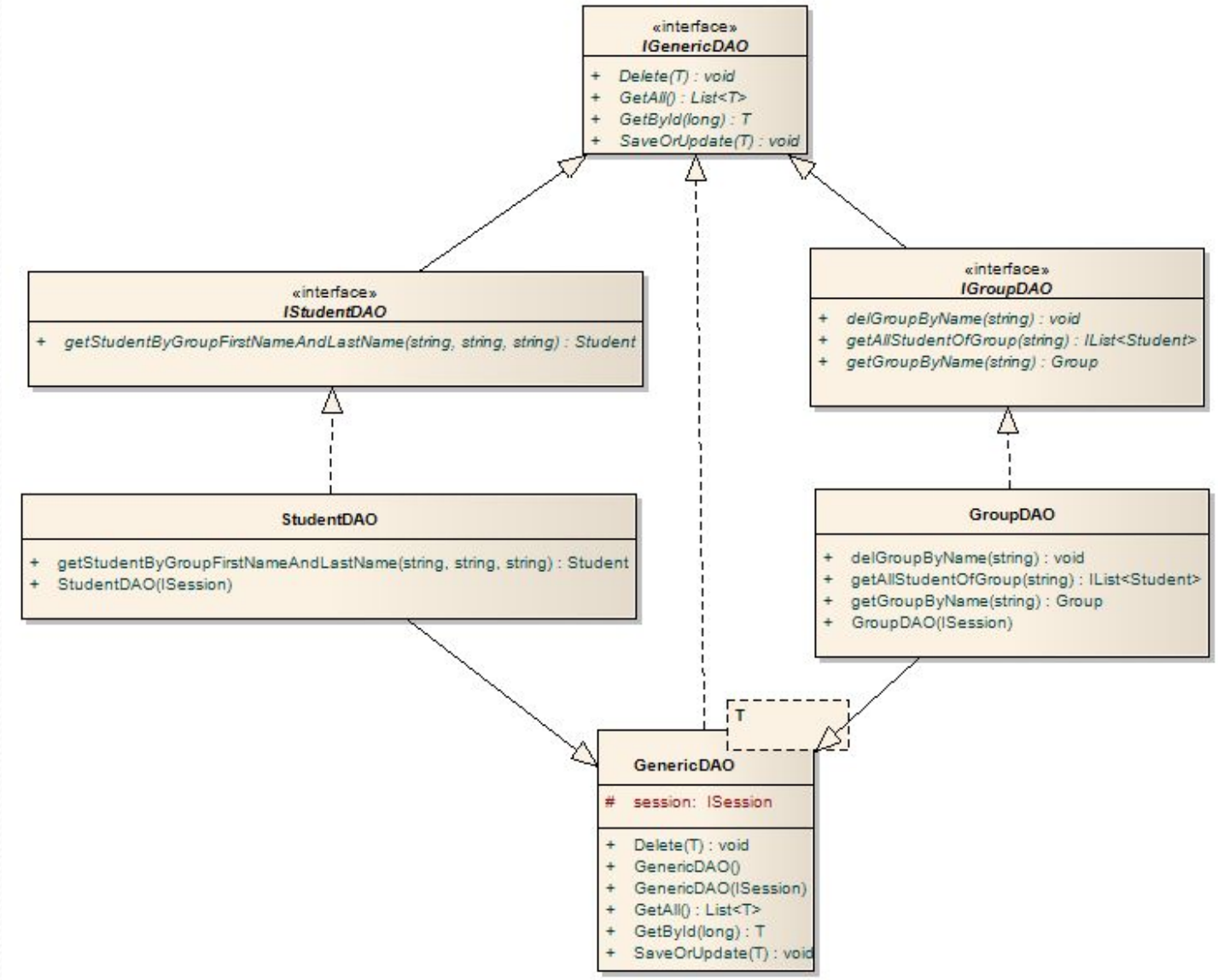
Запити з використанням HQL

```
IQuery query = session.CreateQuery("from Student as stud where stud.Name = :Name");
query.SetAnsiString("Name", "Марія");
List<Student> list = query.List<Student>();
```

```
IQuery query1 = session.CreateQuery(
        "select st.Surname
        from Student as st
        inner join st.Group as gr
        where gr.GroupName = :Name");
query1.SetAnsiString("Name", "КІ-141");
IList<String> list1 = query1.List<String>();
```

```
IQuery query1 = session.CreateQuery(
        "select st.Surname
        from Student as st
        inner join st.Group as gr
        where gr.GroupName like 'КІ-%'");
IList<String> list1 = query1.List<String>();
```

Діаграма класів шаблону «Generic DAO»



Шаблон «Generic DAO»

```
public class GenericDAO<T> : IGenericDAO<T>{
    private ISession session;
    public GenericDAO(ISession session){
        this.session = session;
    }
    public void Save(T item){
        session.Save(item);
    }
    public T ReadByID(long ID){
        return session.Get<T>(ID);
    }
    public List<T> ReadAll(){
        return new
     List<T>(session.CreateCriteria(typeof(T)).List<T>());
    }
    public void Delete(T item){
        session.Delete(item);
    }
}
```

```
public interface IGenericDAO<T>{
    void Save(T item);
    T ReadByID(long ID);
    List<T> ReadAll();
    void Delete(T item);
}
```

Шаблон «Фабрика»