УДК 681.3.06

ПРИМЕНЕНИЕ БИБЛИОТЕК, ОБЕСПЕЧИВАЮЩИХ ДОСТУП К БАЗАМ ДАННЫХ

А.М.МАМБЕТАЛИЕВА
E.mail. ksucta@elcat.kg

Бул макалада NHibernate китепканасын, ошондой эле анын негизинде түзүлгөн Fluent NHibernate жана Castle ActiveRecord китепканаларын пайдалануу жагы каралган.

В статье рассматривается использование библиотек NHibernate, а также библиотек, построенных на ее основе: Fluent NHibernate и Castle ActiveRecord.

This article describes how to use NHibernate libraries and libraries constructed on the basis of its such as Fluent NHibernate and Castle ActiveRecord.

Существует множество определений термина «База данных», однако все они сводятся к понятию совокупности хранимых данных (записей). Данные в базе данных (БД) хранятся в соответствии с моделью данных.

На данный момент самой распространенной является реляционная модель. Реляционная база данных использует набор таблиц, связанных друг с другом посредством ключей[1].

Сейчас на рынке существует множество как коммерческих, так и открытых реляционных СУБД[2]: SQL Server, Oracle, MySQL, PostgreSQL.

Паттерны[3] доступа к данным

После реализации сущностей предметной области возникает необходимость в преобразовании полей и свойств объектов сущностей в поля таблиц БД. Для этой цели используются классы ORM[4], с помощью которых достаточно просто определить отображение свойств объекта на таблицу БД. ORM-библиотека будет генерировать SQL[5] запросы и преобразовывать данные.

При выборе стратегии реализации работы с данными необходимо, в первую очередь, определить, какие классы будут отвечать за отображение. Ниже приведены два существующих варианта.

1. Active Record

Один объект управляет и данными, и поведением. Большинство этих данных постоянны, и их надо хранить в базе данных. Этот паттерн использует наиболее очевидный подход – хранение логики доступа к данным в объекте сущности. Объект является "оберткой" одной строки из БД или представления, включает в себя доступ к базам данных и логику обращения с данными /3/.

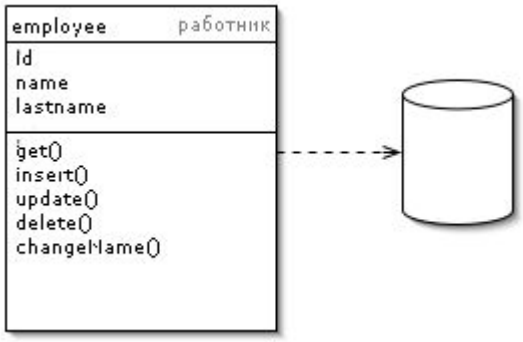

Рис. 1. Паттерн проектирования Active Record

2. Data Mapper

Это программная прослойка, разделяющая объект и БД. Его обязанность – пересылать данные между ними и изолировать их друг от друга. При использовании

Data Mapper'a объекты не нуждаются в знании о существовании БД. Они не нуждаются в SQL-коде и, естественно, в информации о структуре БД. Так как Data Mapper – это разновидность паттерна Mapper, сам объект-Mapper неизвестен объекту /4/.

---

[1] *Ключ,* или *потенциальный ключ* – это минимальный набор атрибутов, по значениям которых можно однозначно выбрать требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждая сущность должна, но не обязана обладать хотя бы одним возможным ключом.

[2] *Система управления базами данных (СУБД)* – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

[3] *Паттерны* (англ. *'pattern* – образец, шаблон, система) – эффективный способ решения характерных задач проектирования.

[4] *ORM (Object Relational Mapper)* – объектно-реляционное отображение.

[5] SQL(Structured Query Language, язык структурированных запросов) – язык, используемый для определения данных, доступа к данным и их обработки.

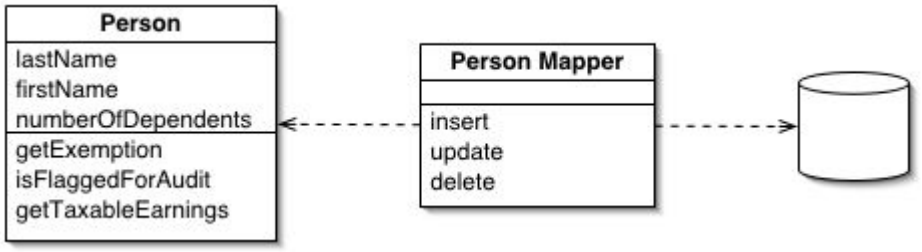

Рис. 2. Паттерн проектирования Data Mapper

Репозиторий (Repository)

Инкапсулирует объекты, представленные в хранилище данных, и операции, производимые над ними, предоставляя более объектно-ориентированное представление реальных данных. Repository также преследует цель достижения полного разделения и односторонней зависимости между уровнями области определения и распределения данных /5/.

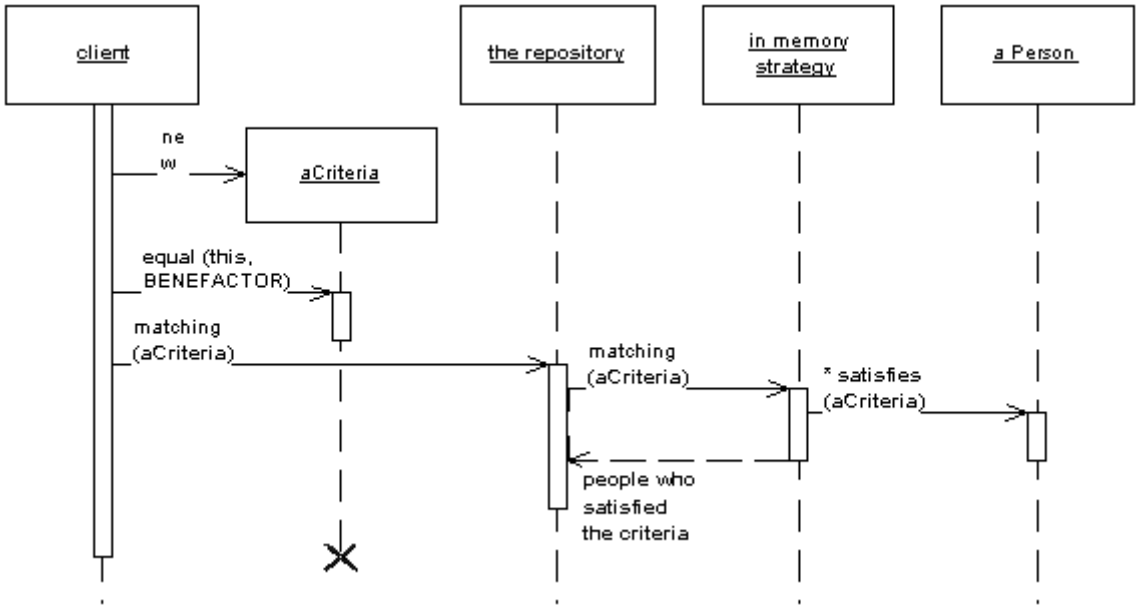

Рис. 3. Паттерн проектирования Repository

Средства доступа к данным в среде .NET[6]

NHibernate

NHibernate – ORM-решение для платформы Microsoft .NET портированное с Java. NHibernate позволяет отображать объекты бизнес-логики на реляционную базу

данных. По заданному XML-описанию сущностей и связей NHibernate автоматически создает SQL-запросы для загрузки и сохранения объектов.

Определим класс предметной области следующим образом:

```
public class Student
    {
        public virtual int Id  { get;    set; }
        public virtual string FullName { get;    set; }
    }

public class Movie
   {
        public virtual int Id  { get;    set; }
        public virtual string Title { get;    set; }
        public virtual DateTime ReleaseDate  { get;    set; }
        public virtual Student Student  { get;    set; }
    }
```

Рассмотрим слой репозиториев.

```
……..
public class StudentRepository: RepositoryBase <Student>
{
public Student FindByName (string name)
   {
 using  (ISession session=NHibernateHelper.OpenSession ())
     {
        ICriteria criteria=session. CreateCriteria <Student> ();
        criteria.Add(Expression.Eq (“FullName”, name ));
        criteria.SetMaxResults(1);
        var result = criteria List <Student> ();
        return result. Count == 0 ? null : criteria. List <Student> ( ) [0];
     }
}
}
```

Таким образом, теперь можно получить список всех записей из таблицы Students следующим образом:

```
var studentRepository=new StudentRepository();
var allStudents=studentRepository.GetAll();
```

Для поиска по имени можно воспользоваться методом FindByName.

```
var  findResult=studentRepository.FindByName (“Some Name”);
```

Fluent Nhibernate

Fluent Nhibernate – это библиотека, позволяющая конфигурировать отображения nhibernate за счет декларативного описания (без использования xml-файлов).

Для вышеприведенного примера конфигурация будет осуществляться следующим образом:

```
public class StudentMap: ClassMap <Student>
{
public StudentMap()
{
Table (“Students”);
```

```
                Id (x => x.Id);
                Map (x =>x.FullName);
                }
}
public class MovieMap : ClassMap <Movie>
{
    public MovieMap()
                {
                        Table (“Movies”);
Id (x =>  x.Id);
```

[6] **.**NET – программная платформа, основой которой является исполняющая среда Common Language Runtime(CLR), способная выполнять как обычные программы, так и серверные веб-приложения.

```
                                Map (x =>x.Title);
                        Map (x =>x.ReleaseDate);
                        References (x=> x.Student).Column (“StudentId”);
                }
}
```

Castle ActiveRecord

Castle ActiveRecord – реализация ActiveRecord поверх NHibernate. Основное достоинство данной библиотеки – это простота использования и настройки.

Следующий вид имеет классы доступа к данным:

```
public  class Student : ActiveRecordBase<Student>
  {
     public int Id { get;  set; }
     public string FullName { get;  set; }
  }
     public class Movie: ActiveRecordBase <Movie>
    {
     public int Id  {get ;  set; }
     public string Title {get;  set; }
     public DateTime  ReleaseTime { get;  set; }
     public Student Student   {get;   set;}
     }
```

Использование приведенных настроек:

```
var allStudents = Student.FindAll ();
```

Заключение

При создании современных приложений корпоративного уровня работа с объектно-ориентированным программным обеспечением и реляционной базой данных может быть достаточно обременительной и трудоемкой.

NHibernate является инструментом объектно-реляционного отображения данных (object/relational mapping tool = ORM tool). Термин объектно-реляционное отображение (object/relational mapping = ORM) относится к технике отображения объектно-ориентированных данных в реляционную модель со схемой данных, основанной на SQL.

NHibernate не только заботится об отображении объектов классов в таблицы базы данных, но также обеспечивает механизм запроса и поиска данных и может значительно сократить время разработки программного продукта.

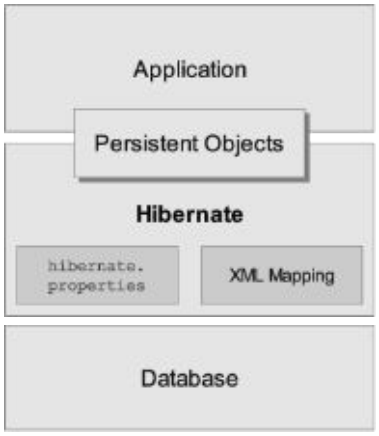

Рис 4. Архитектура NHibernate

NHibernate использует базу данных (Database) и конфигурационные данные (hiberanate.properties, XML mapping) для предоставления сервисов, делающих объекты долгоживущими (Pesistent Objects). Данные сервисы используются приложением (Application) /14/.

Целью Hibernate является освобождение разработчика от 94 процентов общих работ, связанных с задачами программирования долгоживущих (persistence) данных. Возможно, NHibernate не является лучшим решением для приложений, централизованных вокруг данных (data-centric applications), которые используют только хранимые процедуры для реализации бизнес логики в базе данных, но является наиболее полезным при использовании объектно-ориентированных моделей предметной области и бизнес-логики, основанных на промежуточном слое (middle-tier) (например, в языке программирования высокого уровня Java). Тем не менее, NHibernate помогает удалить из приложения или инкапсулировать (скрыть) зависящий от поставщика SQL-код и также помогает в решении стандартной задачи преобразования набора данных (result set) из табличного представления в объектный граф.

С приходом NHibernate появилась возможность описывать объектную модель в виде набора POCO[7]-объектов. Если со стороны БД у нас есть связи, то со стороны .NET у нас будут navigation-свойства с типизированными коллекциями. Меппинг[8] (Mapping) между .NET и БД описывается в специальных *.hbm.xml-файлах. Вполне возможно, этот подход можно было бы назвать удобным, если бы у него была мощная поддержка в Visual Studio в виде визуального дизайнера.

При использовании Castle ActiveRecord должны помечать свои классы и свойства атрибутами, которые описывают меппинг к базе данных, а это нарушает Persistence Ignorance.

Persistence Ignorance означает, что классы данных не имеют статических зависимостей от каких-то специфичных вещей, относящихся только к persistence. А именно не требуется наследовать их от какого-то специального класса из фреймворка или добавлять дополнительные атрибуты на свойства.

В любом случае, подход Castle ActiveRecord вполне возможно использовать в небольших проектах.

Описание меппинга с помощью Fluent NHibernate является наиболее оптимальным. В Fluent NHibernate меппинг описывается на C# языке, что упрощает рефакторинг, предполагает наличие IntelliSense.

Технология IntelliSense предоставляет много удобств, которые делают легко доступными справочники по языку программирования. Во время программирования нет необходимости покидать Редактор кода и текста или Окно интерпретации – окно команд для поиска элементов языка. Можно сохранить контекст, найти необходимые сведения,

вставить элементы языка прямо в код и даже позволить функции IntelliSense завершать ввод /13/.

Также с помощью Fluent NHibernate можно провести минимальную проверку грубых оплошностей в виде compile-time errors (ошибка времени компиляции), которая происходит, когда пишется код с синтаксическими ошибками. Кроме того, у Fluent NHibernate есть две Киллер-фича[9] – это Auto Mapping + Conventions и Persistence specification testing.

Список литературы

1. Мартин Фаулер. Архитектура корпоративных программных приложений. – М.: Вильямс, 2006.

[7] *POCO (Plain old CLR object)* – это коллекция библиотек классов с открытым исходным кодом, которая упрощает и ускоряет разработку сетевых мультиплатформенных приложений.

[8] *Меппинг* (от англ. mapping – нанесение на карту, отображение) – процесс составления схемы того, какими данными следует обмениваться, как они будут использоваться и каким системам управления они нужны.

[9] Киллер-фича (от англ. *Killer-feature* – убийственная особенность) – определенная особенность или черта программного продукта, которая выделяет его на фоне конкурентов.

2. Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2006.
3. http://design-pattern.ru/patterns/active-record.html
4. http://design-pattern.ru/patterns/data-mapper.html
5. http://design-pattern.ru/patterns/repository.html
6. http://sourceforge.net/projects/nhibernate/files/NHibernate/3.3.1GA/NHibernate-3.3.1.GA-bin.zip/download
7. http://fluentnhibernate.org/
8. http://castleproject.org/ActiveRecord/
9. Джимми Нильсон. Применение DDD и шаблонов проектирования. Проблемно-ориентированное программирование на C# и .NET. – М.: Вильямс, 2008.
10. Eric Evans. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional, 2003.
11. Pierre Henri Kuate, Tobin Harris, Christian Bauer, and Gavin King. Nhibernate in Action. Manning publications,2009. 400c.
12. Jason Dentler.Nhibernate 3.0.Cookbook. Packt publishing, 2010. 328c.
13. http://msdn.microsoft.com/ru-ru/library/hcw1s69b(v=vs.90).aspx
14. http://samsonov.bn.by/lib/hibernate/architecture.html#architecture-overview