

itransition  
**knowledge**  
center

## ORM: NHibernate

Иван Позняк

# 1. План

- ORM
- NHibernate: Basics
- NHibernate: Advanced
- Что дальше?

A large red circle is positioned in the upper right quadrant of the page. Inside the circle, the text "itransition knowledge center" is written in white. "itransition" is in a smaller, lowercase sans-serif font. "knowledge" is in a larger, bold, lowercase sans-serif font. "center" is in a smaller, lowercase sans-serif font, positioned below "knowledge".

itransition  
**knowledge**  
center

ORM

## 3. ORM

- Object
- Relational
- Mapping
- реляционная БД □ □ язык

программирования

## 4. ORM только на .NET / Java? – нет!

- C++ / Java / .NET / Flex / Delphi / Objective-C /

Perl / PHP / Python / Ruby и даже VisualBasic 6.0

- Альтернатива – no SQL решения

## 5. Пример проекта

- 10 классов-моделей
- Необходимо CRUD приложение
- Некоторые сущности связаны
- Некоторые поля сущностей «сложные»
- В будущем нужен репортинг

## 6. Вариант 1 – не используем ORM

- ... на самом деле пишем «свой»
- У нас больше контроля
- Но мы пишем больше кода
- Больше шансов ошибиться

## 7. Вариант 2 – используем ORM

- Используем уже существующие решения
- Ограничены возможностями библиотеки
- Пишем больше «конфигурации»
- Пишем меньше кода
- Меньше шансов ошибиться



## 8. Вывод

- Не использовать готовое – не вариант
- Вариант – использовать лучшее из готового
- Чистый SQL никто не отменяет



iTransition  
**Knowledge**  
center

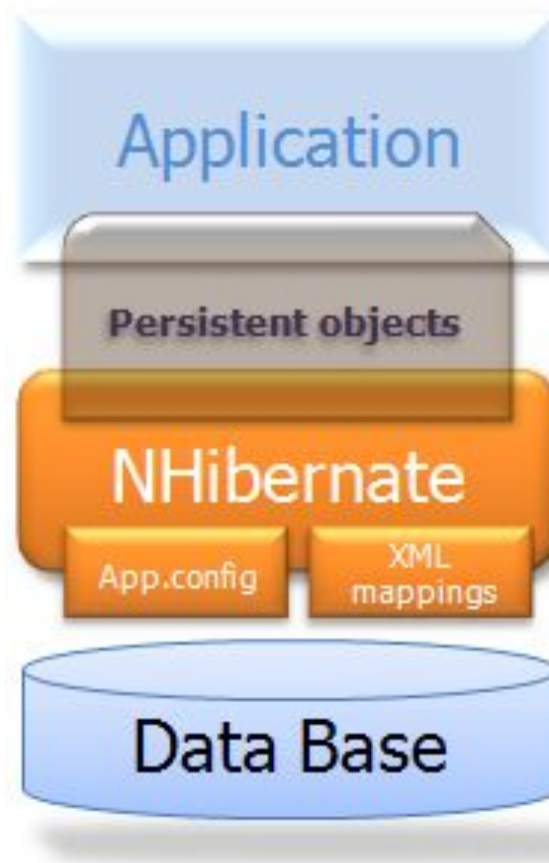
NHibernate

## 10. Кратко о библиотеке

- Оригинальная Java версия – Hibernate
- Версия 3.2.0
- Сайт <http://nhforge.org>

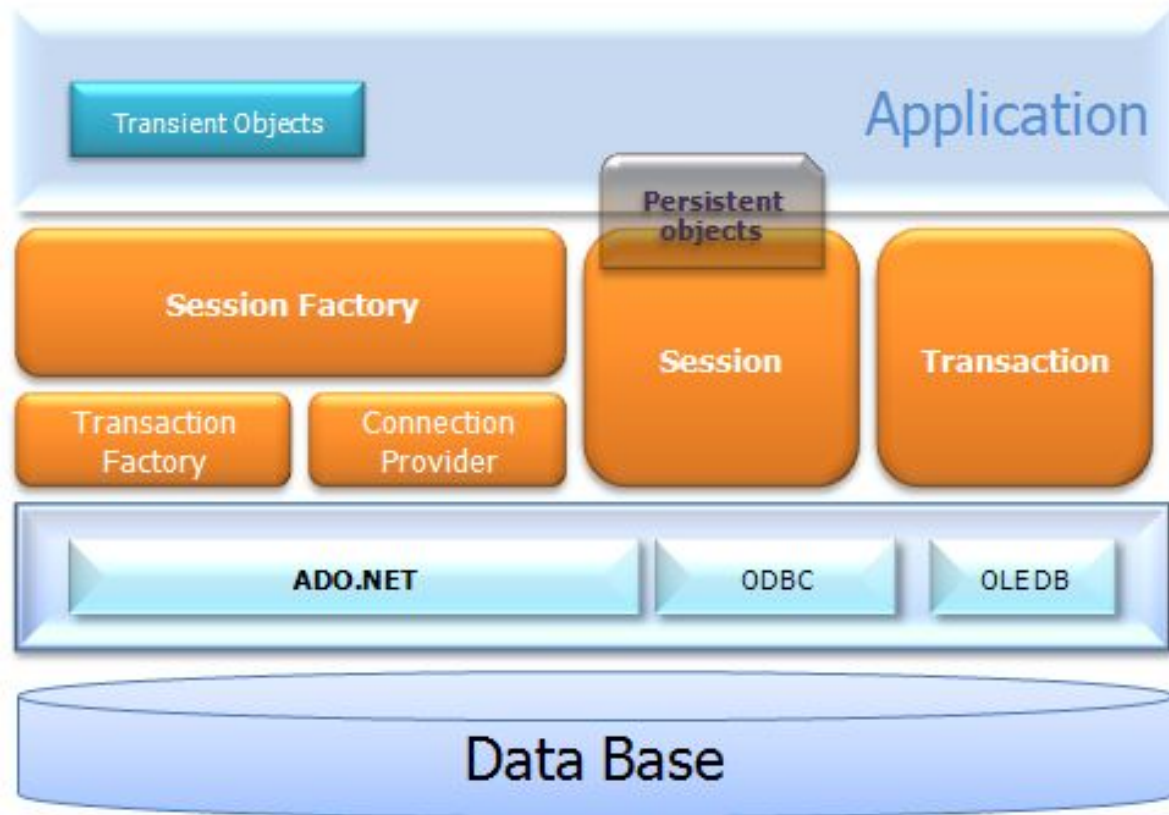
# 11. Как это работает

- Краткая версия



# 12. Как это работает

- Полная версия



# 13. СУЩНОСТИ И ИХ СОСТОЯНИЯ

- transient
- persistent
- detached

# 14. Сами сущности

- POJO classes
- Свойства вместо полей
- Конструктор по умолчанию
- Видимость не важна
- Equals / GetHashCode

## 15. Динамические модели

- Можно обойтись без классов моделей
- Dictionary<String, Object>
- В мэппингах – entity-name у класса
- В конфиге - default\_entity\_mode
- В коде – у сессии EntityMode.Map



A large red circle is positioned in the upper right quadrant of the slide. Inside the circle, the text "itransition knowledge center" is written in white. "itransition" is in a smaller, lowercase sans-serif font. "knowledge" is in a larger, bold, lowercase sans-serif font. "center" is in a smaller, lowercase sans-serif font, positioned below "knowledge".

itransition  
**knowledge**  
center

Callbacks

## 17. Жизненный цикл

- `interface ILifecycle`
- `void OnSave (ISession s, object id);`
- `LifecycleVeto OnSave (ISession s);`
- `LifecycleVeto OnUpdate (ISession s);`
- `LifecycleVeto OnDelete (ISession s);`

## 18. Валидация

- IValidatable
- void Validate();
- ValidationFailure exception
- Время вызова – не гарантируется

A large red circle is positioned in the upper right quadrant of the slide. Inside the circle, the text 'itransition knowledge center' is written in white. 'itransition' is in a smaller, lowercase sans-serif font. 'knowledge' is in a larger, bold, lowercase sans-serif font. 'center' is in a smaller, lowercase sans-serif font, positioned below 'knowledge'.

itransition  
**knowledge**  
center

Mapping

## 20. Mapping through XML

- Расширение .hbm.xml
- Embedded resource
- Регистрация через конфигурационный файл

## 21. Дополнительные объекты БД

- В mapping файле
- В классе реализующем

`NHibernate.Mapping.IAuxiliaryDatabaseObject`

- Можно параметризовать
- Можно выполнять только для

определенных диалектов БД

```
<nhibernate-mapping>
  <database-object>
    <create>CREATE TRIGGER my_trigger ...</create>
    <drop>DROP TRIGGER my_trigger</drop>
  </database-object>
</nhibernate-mapping>
```

```
<hibernate-mapping>
  <database-object>
    <definition class="MyTriggerDefinition, MyAssembly">
      <param name="parameterName">parameterValue</param>
    </definition>
  </database-object>
</hibernate-mapping>
```

```
<hibernate-mapping>
  <database-object>
    <definition class="MyTriggerDefinition"/>
    <dialect-scope name="NHibernate.Dialect.Oracle9Dialect"/>
    <dialect-scope name="NHibernate.Dialect.OracleDialect"/>
  </database-object>
</hibernate-mapping>
```

## 23. Mapping Fluent NHibernate (3.1)

- Пишется код для формирования mapping
- Конфигурируется тоже через код
- Примеры



Традиционный mapping:

```
<?xml version="1.0" encoding="utf-8" ?>
  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
namespace="QuickStart" assembly="QuickStart">
  <class name="Cat" table="Cat">
    <id name="Id">
      <generator class="identity" />
    </id>
    <property name="Name">
      <column name="Name" length="16" not-null="true" />
    </property>
    <property name="Sex" />
    <many-to-one name="Mate" />
    <bag name="Kittens">
      <key column="mother_id" />
      <one-to-many class="Cat" />
    </bag>
  </class>
</hibernate-mapping>
```

## Конфигурация через hibernate.cfg.xml

```
<?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="connection.driver_class">
      NHibernate.Driver.SqlClientDriver
    </property>
    <property name="connection.connection_string">
      Server=localhost;initial catalog=nhibernate;User Id=;Password=
    </property>
    <property name="show_sql">>false</property>
    <property name="dialect">NHibernate.Dialect.MsSql2000Dialect</property>

    <mapping
      resource="NHibernate.Auction.Item.hbm.xml" assembly="NHibernate.Auction" />
    <mapping
      resource="NHibernate.Auction.Bid.hbm.xml" assembly="NHibernate.Auction" />

  </session-factory>
</hibernate-configuration>
```

Mapping через Fluent Nhibernate:

```
public class CatMap : ClassMap<Cat>
{
    public CatMap()
    {
        Id(x => x.Id);
        Map(x => x.Name).Length(16).Not.Nullable();
        Map(x => x.Sex);
        References(x => x.Mate);
        HasMany(x => x.Kittens);
    }
}
```

Конфигурация:

```
Fluently.Configure().Database(SQLiteConfiguration
    .Standard
    .UsingFile("firstProject.db"))
.Mappings(m => m.FluentMappings.AddFromAssemblyOf<Program>())
.BuildSessionFactory();
```

## 27. Mapping via code NHibernate (3.2)

- Пишется код для формирования mapping
- Конфигурируется тоже через код
- Примеры

```
ModelMapper mapper = new ModelMapper();
mapper.Class<Person>(m =>
{
    m.Id(k => k.Id, g=>g.Generator(Generators.Native));
    m.Table("People");
    m.Property(k => k.Name);
    m.Bag(k => k.Addresses,
t => { t.Table("PeopleAddresses"); t.Key(c=>c.Column("PersonId"));
        t.Inverse(true); },
rel => rel.ManyToMany(many => many.Column("AddressId")) );
});
```

```
mapper.Class<Address>(m =>
{
    m.Id(k => k.Id, g => g.Generator(Generators.Native));
    m.Table("Addresses");
    m.Property(p => p.City);
    m.Join("PeopleAddresses", z =>
{
    z.Property(p => p.IsDefault);
    z.Property(p => p.ValidFrom);
    z.Property(p => p.ValidTo);
    z.Key(k => k.Column("PersonId"));
});
});
```

```
});
```



itransition  
**knowledge**  
center

Associations

## 30. One-to-One

- Primary-key based
- `<one-to-one ... />`
- Foreign-key based
- `<many-to-one ... unique="true" />`

## 31. One-to-Many

- <set> - ISet
- <list> - IList
- <map> - IDictionary
- <bag> - IList
- <array> и <primitive-array> - []
- В коллекциях – сущности / простые типы



## 32. Many-to-One

- <many-to-one>

## 33. Many-to-Many

- <set>

- <bag>

- <many-to-many>

## 34. Bi-directional associations

- Два разных представления в памяти
- Возможные проблемы решаются с

помощью `inverse="true"`

- `inverse <=/=> cascade`

## 35. Cascade

- all
- save-update
- delete
- all-delete-orphan



iTransition  
**Knowledge**  
center

Lazy initialization

## 37. Механизм

- Возвращается прокси а не сам объект
- Данные загружаются только по

необходимости

- Возникают проблемы если сессия закрыта
- По умолчанию включен для классов
- Два подхода – класс наследник, либо

## 38. Ограничения

- Класс не sealed
- Свойства и методы – virtual
- Конструктор по умолчанию не private
- Если проху строится на базе интерфейса –  
ограничения не действуют

```
s = sessions.OpenSession();
```

```
User u = (User) s.Find("from User u where u.Name=?", userName,  
NHibernateUtil.String)[0];
```

```
IDictionary permissions = u.Permissions;
```

```
s.Close();
```

```
...
```

```
int accessLevel = (int) permissions["accounts"]; // Error!
```



## 40. Lazy initialization и коллекции

- Можно отключать
- Атрибут lazy у коллекций
- Не усердствовать с lazy="false"

## 41. LazyInitializationException

- Не закрывать сессию (до завершения обработки запроса)
- Вызвать руками `NHibernateUtil.Initialize()` передав туда коллекцию
- Привязать объект к сессии через `Update / Lock`

Вариант получения общего кол-ва элементов в коллекции не загружая ее из базы:

```
ICollection countColl = s.Filter( collection, "select count(*)" );
```

```
IEnumerator countEn = countColl.GetEnumerator();
```

```
countEn.MoveNext();
```

```
int count = (int) countEn.Current;
```



iTransition  
**Knowledge**  
center

Inheritance

## 44. 3 стратегии

- Table per hierarchy
- Table per subclass
- Table per class
- Если в иерархии есть абстрактные классы –

`abstract=true`

```
<class name="IPayment" table="PAYMENT">
  <id name="Id" type="Int64" column="PAYMENT_ID">
    <generator class="native"/>
  </id>
  <discriminator column="PAYMENT_TYPE" type="String"/>
  <property name="Amount" column="AMOUNT"/>
  ...
  <subclass name="CreditCardPayment" discriminator-value="CREDIT">
    ...
  </subclass>
  <subclass name="CashPayment" discriminator-value="CASH">
    ...
  </subclass>
  <subclass name="ChequePayment" discriminator-value="CHEQUE">
    ...
  </subclass>
</class>
```

```
<class name="IPayment" table="PAYMENT">
  <id name="Id" type="Int64" column="PAYMENT_ID">
    <generator class="native"/>
  </id>
  <property name="Amount" column="AMOUNT"/>
  ...
  <joined-subclass name="CreditCardPayment" table="CREDIT_PAYMENT">
    <key column="PAYMENT_ID"/>
    ...
  </joined-subclass>
  <joined-subclass name="CashPayment" table="CASH_PAYMENT">
    <key column="PAYMENT_ID"/>
    ...
  </joined-subclass>
  <joined-subclass name="ChequePayment" table="CHEQUE_PAYMENT">
    <key column="PAYMENT_ID"/>
    ...
  </joined-subclass>
</class>
```

```
<id name="id" type="int64" column="PAYMENT_ID" >
  <generator class="native"/>
</id>
<discriminator column="PAYMENT_TYPE" type="string"/>
<property name="Amount" column="AMOUNT"/>
...
<subclass name="CreditCardPayment" discriminator-value="CREDIT">
  <join table="CREDIT_PAYMENT">
    <key column="PAYMENT_ID"/>
    <property name="CreditCardType" column="CCTYPE"/>
    ...
  </join>
</subclass>
<subclass name="CashPayment" discriminator-value="CASH">
  <join table="CASH_PAYMENT">
    <key column="PAYMENT_ID"/>
    ...
  </join>
</subclass>
<subclass name="ChequePayment" discriminator-value="CHEQUE">
  <join table="CHEQUE_PAYMENT" fetch="select">
    <key column="PAYMENT_ID"/>
    ...
  </join>
</subclass>
```



```
<class name="Payment">
  <id name="Id" type="Int64" column="PAYMENT_ID">
    <generator class="sequence"/>
  </id>
  <property name="Amount" column="AMOUNT"/>
  ...
  <union-subclass name="CreditCardPayment" table="CREDIT_PAYMENT">
    <property name="CreditCardType" column="CCTYPE"/>
    ...
  </union-subclass>
  <union-subclass name="CashPayment" table="CASH_PAYMENT">
    ...
  </union-subclass>
  <union-subclass name="ChequePayment" table="CHEQUE_PAYMENT">
    ...
  </union-subclass>
</class>
```

```
<class name="CreditCardPayment" table="CREDIT_PAYMENT">
  <id name="Id" type="Int64" column="CREDIT_PAYMENT_ID">
    <generator class="native"/>
  </id>
  <property name="Amount" column="CREDIT_AMOUNT"/>
  ...
</class>
```

```
<class name="CashPayment" table="CASH_PAYMENT">
  <id name="Id" type="Int64" column="CASH_PAYMENT_ID">
    <generator class="native"/>
  </id>
  <property name="Amount" column="CASH_AMOUNT"/>
  ...
</class>
```

```
<class name="ChequePayment" table="CHEQUE_PAYMENT">
  <id name="Id" type="Int64" column="CHEQUE_PAYMENT_ID">
    <generator class="native"/>
  </id>
  <property name="Amount" column="CHEQUE_AMOUNT"/>
  ...
</class>
```

A large red circle is positioned in the upper right quadrant of the page. Inside the circle, the text "itransition knowledge center" is written in white. "itransition" is in a smaller, lowercase sans-serif font. "knowledge" is in a larger, bold, lowercase sans-serif font. "center" is in a smaller, lowercase sans-serif font, positioned below "knowledge".

itransition  
**knowledge**  
center

Operations

## 51. Загрузка по Id

- `.Load(object, id)`
- `.Load(id)`
- `.Get(id)`
- `Load` – пробросит exception
- `Get` – вернет null

## 52. Save / Update / Delete

- Save ~ INSERT
- Update ~ UPDATE
- Delete ~ DELETE
- SaveOrUpdate ~ INSERT || UPDATE
- Для persistent объектов в рамках их

контекста можно ничего дополнительно не

## 53. Критерии

- `Session.CreateCriteria<...>`
- `ICriteria.Add( Restrictions. ...)`
- Restrictions: `Eq / Gt / Lt / In ...`
- Но лучше – `QueryOver` API
- `Session.QueryOver<...>`

С использованием критериев:

```
.Add(Restrictions.And(  
    Restrictions.Eq("Name", "test name"),  
    Restrictions.Or(  
        Restrictions.Gt("Age", 21),  
        Restrictions.Eq("HasCar", true))))
```

С использованием QueryOver

```
.Where(p => p.Name == "test name" && (p.Age > 21 || p.HasCar))
```

```
session.QueryOver<Cat>()  
    .JoinQueryOver<Kitten>(c => c.Kittens)  
    .Where(k => k.Name == "Tiddles");
```

```
.Where(p => p.BirthDate.YearPart() == 1971)
```

```
.Select( p => Projections.Concat(p.LastName, ", ", p.FirstName), p  
=> p.Height.Abs())
```

## 55. Fetch Mode

- User user = (User)

```
session.CreateCriteria(typeof(User))
```

```
.SetFetchMode("Permissions", FetchMode.Join)
```

```
.Add( Restrictions.Eq("Id", userId) )
```

```
.UniqueResult();
```



## 56. Batch Size

- На коллекциях и классах в mapping файлах

## 57. Second Level Cache

- Read only
- Read / Write
- Non-strict Read / Write

## 58. Query Cache

- `.SetCacheable(true)`
- `.SetCacheableRegion("")`
- Для сброса кэша
- `.SetForceCacheRefresh(bool)`
- `ISessionFactory.EvictQueries()`

## 59. Multi Criteria & Queries

- Можно выполнить несколько SQL запросов

за один заход к серверу

- Весьма полезно при реализации paging

```
IMultiCriteria multiCrit = s.CreateMultiCriteria()  
    .Add(s.CreateCriteria(typeof(Item))  
        .Add(Expression.Gt("Id", 50))  
        .SetFirstResult(10))  
    .Add(s.CreateCriteria(typeof(Item))  
        .Add(Expression.Gt("Id", 50))  
        .SetProject(Projections.RowCount()));  
IList results = multiCrit.List();  
IList items = (IList)results[0];  
long count = (long)((IList)results[1])[0];
```

## 61. Работа с сессиями

- Можно задать настройку

`hibernate.current_session_context_class`

- Теперь можно использовать

`ISessionFactory.GetCurrentSession()`

## 62. Генерация схемы

- Configuration cfg = ....;
- new SchemaExport(cfg).Create(false, true);

Further Reading:

<http://nhforge.org/doc/nh/en/index.html> - документация

<http://nhforge.org/blogs/nhibernate/default.aspx> - блог



IKC@itransition.com  
@ItransitionKC

A large red circle is positioned in the lower right quadrant of the page, partially overlapping the white background. It contains the organization's name in white text.

itransition  
**knowledge**  
center