CriteriaLoader – класс по формированию критериев для загрузки со скриптом

- loaders[0] {NHibernate.Loader.Criteria.CriteriaLoader(SELECT this\_.CustomerId as CustomerId0\_0\_, this\_.FirstName as FirstName0\_0\_, this\_.LastName as LastName0\_0\_ FROM Customer this\_)} NHibernate.Loader.Criteria.CriteriaLoader

+ ((NHibernate.Loader.OuterJoinLoader)(loaders[0])).sql

{SELECT this\_.CustomerId as CustomerId0\_0\_, this\_.FirstName as FirstName0\_0\_, this\_.LastName as LastName0\_0\_ FROM Customer this\_}

NHibernate.SqlCommand.SqlString

Вопрос по классу Nhibernate.ArrayHelper

Базовая цепочка событий

+ results Count = 0 System.Collections.IList {System.Collections.Generic.List<ItCuties.NHibernate.Customer>}

Важный вопрос 0 tx.CommitTransaction

FlushEvent – сохранение данных

flushEventListener[i].OnFlush(new FlushEvent(this));

IDbTransaction : IDisposable

var tx = session.BeginTransaction() – AdoTransaction

Получение данных Session.List

Класс SessionIml List

Анализ логики чтения данных.

1)Создание конфигурации

var cfg = new Configuration(); - создание объекта конфигурации.

cfg.DataBaseIntegration – вызов метода по настройке интеграции с БД.

ConfigurationExtensions

Делегат – объект, который определяется сигнатурой метода и возвращаемым типом.

Далее к нему могут подключаться ссылки на методы,

которым он будет делегировать выполнение действий с объектами, которые

происходят при вызове сигнатуры базового определенного для него метода.

Получение данных:

public override void List(CriteriaImpl criteria, IList results)

{

using (new SessionIdLoggingContext(SessionId))

{

CheckAndUpdateSessionStatus();

string[] implementors = Factory.GetImplementors(criteria.EntityOrClassName);

int size = implementors.Length;

CriteriaLoader[] loaders = new CriteriaLoader[size];

ISet<string> spaces = new HashSet<string>();

for (int i = 0; i < size; i++)

{

loaders[i] = new CriteriaLoader(

GetOuterJoinLoadable(implementors[i]),

Factory,

criteria,

implementors[i],

enabledFilters

);

spaces.UnionWith(loaders[i].QuerySpaces);

}

AutoFlushIfRequired(spaces);

dontFlushFromFind++;

bool success = false;

try

{

for (int i = size - 1; i >= 0; i--)

{

ArrayHelper.AddAll(results, loaders[i].List(this));

}

success = true;

}

catch (HibernateException)

{

// Do not call Convert on HibernateExceptions

throw;

}

catch (Exception sqle)

{

throw Convert(sqle, "Unable to perform find");

}

finally

{

dontFlushFromFind--;

AfterOperation(success);

}

}

}