Данный проект является чисто исследовательским использования Code First и Linq to EF на примере 1С версии 7.7. Так как сам я программист 1С, то мне всегда было интересно как можно перенести модель объектов 1С на компилируемые языки, и использовать мощь Linq to EF.

С появлением Code First давно хотел прикрутить, но все как то руки не доходили и вот наконец ..

Для начала создадим базовый класс

public class СправочникПредок

{

public virtual string ID { get; set; }

public virtual string Наименование { get; set; }

public virtual bool ПометкаУдаления { get; set; }

public virtual byte ISFOLDER { get; set; }

public virtual object ПолучитьКод() { return ""; }

public virtual string Вид() { return ""; }

public virtual bool ЭтоГруппа() { return ISFOLDER == 1; }

}

Зачем это нужно поясню чуть позже.

На его основе можно создать описание справочника, наприме Номенклатуры

[Table("SC84")]

public partial class Номенклатура : СправочникПредок

{

public DateTime ДатаДляПериодическихРеквизитов = DateTime.Now;

[Key]

[Required]

[StringLength(9)]

override public string ID { get; set; }

[Column("CODE")]

[Required]

[StringLength(8)]

public string Код { get; set; }

[Column("DESCR")]

[Required]

[StringLength(99)]

override public string Наименование { get; set; }

[Column("PARENTID")]

[Required]

[StringLength(9)]

public string РодительId { get; set; }

override public byte ISFOLDER { get; set; }

[Column("ISMARK")]

override public bool ПометкаУдаления { get; set; }

..............................................................

[Column("SP94")]

[Required]

[StringLength(9)]

public string ОсновнаяЕдиницаId { get; set; }

[Column("SP8906")]

[Required]

[StringLength(9)]

public string МатериалId { get; set; }

..............................................................

public virtual object ПолучитьКод() { return Код; }

virtual public Справочник.Номенклатура Родитель { get; set; }

virtual public Справочник.Номенклатура Материал { get; set; }

[InverseProperty("Материал")]

public ICollection<Справочник.Номенклатура> ПодчиненныеДляМатериал { get; set; }

[InverseProperty("Владелец")]

public ICollection<Справочник.Единицы> ПодчиненныеЕдиницы { get; set; }

override public string Вид() { return ВидыСправочников.Номенклатура; }

Чезер атрибуты мы помечаем поля так как нам нужно, при этом создаем свойства нужного нам типа.

Про соглашение имен подробно написано здесь

<http://metanit.com/sharp/entityframework/2.6.php>

Дополнительно обращу ваше внимание на

virtual public Справочник.Номенклатура Материал { get; set; }

[InverseProperty("Материал")]

public ICollection<Справочник.Номенклатура> ПодчиненныеДляМатериал { get; set; }

Это требует Code First для разрешения ссылок когда один тип ссылается на сам себя и описания один ко многим. Также нужно описать коллекции на подчинённые справочники

public partial class Единицы : СправочникПредок

{

public DateTime ДатаДляПериодическихРеквизитов = DateTime.Now;

[Key]

[Required]

[StringLength(9)]

override public string ID { get; set; }

[NotMapped]

override public string Наименование { get { return ""; } set { } }

[Column("PARENTEXT")]

[Required]

[StringLength(9)]

public string ВладелецId { get; set; }

[NotMapped]

override public byte ISFOLDER { get { return 2; } set { } }

[Column("ISMARK")]

override public bool ПометкаУдаления { get; set; }

[Column("SP79")]

[Required]

[StringLength(9)]

public string ОКЕИId { get; set; }

[Column("SP76", TypeName = "numeric")]

public decimal Вес { get; set; }

[Column("SP78", TypeName = "numeric")]

public decimal Коэффициент { get; set; }

[Column("SP80")]

[Required]

[StringLength(13)]

public string ШтрихКод { get; set; }

[Column("SP8752", TypeName = "numeric")]

public decimal Объем { get; set; }

[Column("SP9519")]

[Required]

[StringLength(36)]

public string Ref { get; set; }

virtual public Справочник.Номенклатура Владелец { get; set; }

virtual public Справочник.ОКЕИ ОКЕИ { get; set; }

override public string Вид() { return ВидыСправочников.Единицы; }

}

На этом описательная часть закончена перейдём к самому вкусному.

Используя предка можно написать обобщенную функцию.

public TEntity ПолучитьЭлементСправочника<TEntity>(string ID) where TEntity : СправочникПредок

{

var query2 = from спр in this.Set<TEntity>()

where спр.ID == ID

select спр;

return query2.SingleOrDefault<TEntity>();

}

Теперь мы можем вызвать её таким способом

public static СправочникПредок ПолучитьЗначениеНеопределенногоСправочника(string НеопределенныйИД)

{

var тип = НеопределенныйИД.Substring(0, 4);

var id = НеопределенныйИД.Substring(4, 9);

switch (тип)

{

case ВыдСправочника36.Аналоги:

return БД.ПолучитьЭлементСправочника<Справочник.Аналоги>(id);

case ВыдСправочника36.Банки:

return БД.ПолучитьЭлементСправочника<Справочник.Банки>(id);

При получении неопределенного справочника нам доступны все свойства и методы предка.

А теперь посмотри на мощь LINQ.

В Linq есть два синтаксиса запросов которые можно совмещать

using (var db = new Model1())

{

var Товары = db.Спр\_Номенклатура.Include("ПодчиненныеЕдиницы").Where(p => p.ISFOLDER == 2 && p.РодительId != " 0 " && p.ПодчиненныеЕдиницы.Any()).Take(10);

foreach (var Товар in Товары)

{

Console.WriteLine("{0}.{1} - {2}", Товар.ID, Товар.Наименование, Товар.Код);

foreach (var Единица in Товар.ПодчиненныеЕдиницы)

Console.WriteLine("{0}.{1} - {2}", Единица.ОКЕИ.Наименование, Единица.ШтрихКод, Единица.Коэффициент);

}

}

Такой запрос генерирует запрос к базе не буду указывать все поля укажу только условия

) AS [Project1]

WHERE (2 = [Project1].[ISFOLDER]) AND (N' 0 ' <> [Project1].[PARENTID]) AND ( EXISTS (SELECT

1 AS [C1]

FROM [dbo].[SC75] AS [Extent2]

WHERE [Project1].[ID] = [Extent2].[PARENTEXT]

)) ) AS [Limit1]

LEFT OUTER JOIN [dbo].[SC75] AS [Extent3] ON [Limit1].[ID] = [Extent3].[PARENTEXT]

Так как по умолчанию применяется ленивая загрузка то для получения подчиненных единиц генерируется отдельный запрос к при первом обращении

SELECT

[Extent1].[ID] AS [ID],

[Extent1].[CODE] AS [CODE],

[Extent1].[DESCR] AS [DESCR],

[Extent1].[ISMARK] AS [ISMARK],

[Extent1].[SP42] AS [SP42],

[Extent1].[SP9525] AS [SP9525]

FROM [dbo].[SC41] AS [Extent1]

WHERE [Extent1].[ID] = @EntityKeyValue1

-- EntityKeyValue1: ' 1 ' (Type = String, IsNullable = false, Size = 9)

Перейдем к более сложным **коррелирующим** запросам

var бд = Константы1С.ГлобальныйКонтекст.БД;

var query = from Константа in бд.ТаблицаКонстанты

where Константа.ID == 9697

orderby Константа.DATE descending, Константа.TIME descending, Константа.DOCID descending, Константа.ROW\_ID descending

select Константа;

// select new Константы1С.ЗначениеПериодического { Значение = Константа.VALUE }).Take(1);

var query2= (from спр in бд.Спр\_ДляПериодических

select new

{

Наименование=спр.Наименование,

ДатаСпр=спр.ДатаСпр,

Периодические=(from прериод in query.Where(х=> х.OBJID==спр.ID && х.DATE<=спр.ДатаСпр).Take(1)

select new

{

Значение=прериод.VALUE,

Дата=прериод.DATE

}).FirstOrDefault()

}

);

foreach (var элем in query2)

{

Console.WriteLine("{0}.{1} - {2}", элем.Наименование, элем.ДатаСпр, элем.Периодические);

}

Здесь мечта каждого программиста на 8 ке и 7 ке выполняет следующий код

SELECT

1 AS [C1],

[Extent1].[DESCR] AS [DESCR],

[Extent1].[SP9700] AS [SP9700],

[Limit1].[ROW\_ID] AS [ROW\_ID],

[Limit1].[VALUE] AS [VALUE],

[Limit1].[DATE] AS [DATE]

FROM [dbo].[SC9691] AS [Extent1]

OUTER APPLY (SELECT TOP (1) [Project1].[ROW\_ID] AS [ROW\_ID], [Project1].[DATE] AS [DATE], [Project1].[VALUE] AS [VALUE]

FROM ( SELECT

[Extent2].[ROW\_ID] AS [ROW\_ID],

[Extent2].[DATE] AS [DATE],

[Extent2].[VALUE] AS [VALUE],

[Extent2].[DOCID] AS [DOCID],

[Extent2].[TIME] AS [TIME]

FROM [dbo].[\_1SCONST] AS [Extent2]

WHERE (9697 = [Extent2].[ID]) AND ([Extent2].[OBJID] = [Extent1].[ID]) AND ([Extent2].[DATE] <= [Extent1].[SP9700])

) AS [Project1]

ORDER BY [Project1].[DATE] DESC, [Project1].[TIME] DESC, [Project1].[DOCID] DESC, [Project1].[ROW\_ID] DESC ) AS [Limit1]

Следуе заметить, что если не использовать FirstOrDefault()

То в Периодические будет коллекция.

Можно применять такие запросы

var бд = Константы1С.ГлобальныйКонтекст.БД;

var рез= ( from спр in бд.Спр\_Номенклатура

where спр.ВидНоменклатуры==Перечисление.ВидыНоменклатуры.Услуга

select спр).Take(10);

Для тестов с периодическими реквизитами создал тестовый справочник.

Доступ к ним такой.

[NotMapped]

public DateTime ПериодДата

{

get

{

return Константы1С.Константы.ДатаДляПериодического(9693, ID, ДатаДляПериодическихРеквизитов);

}

}

[NotMapped]

public decimal ПериодЧисло

{

get

{

var query = Константы1С.Константы.ЗапросДляПериодическогоЗначения(9694, ID, ДатаДляПериодическихРеквизитов);

var res = query.SingleOrDefault();

if (res == null)

return 0M;

return decimal.Parse(res.Значение, System.Globalization.CultureInfo.InvariantCulture);

}

}

[NotMapped]

public Int64 ПериодИнт

{

get

{

var query = Константы1С.Константы.ЗапросДляПериодическогоЗначения(9695, ID, ДатаДляПериодическихРеквизитов);

var res = query.SingleOrDefault();

if (res == null)

return 0;

return Int64.Parse(res.Значение, System.Globalization.CultureInfo.InvariantCulture);

}

}

[NotMapped]

public Справочник.Номенклатура ПериодСПР

{

get

{

var query = Константы1С.Константы.ЗапросДляПериодическогоЗначения(9696, ID, ДатаДляПериодическихРеквизитов);

var db = Константы1С.ГлобальныйКонтекст.БД;

var query2 = from спр in db.Спр\_Номенклатура

join конст in query on спр.ID equals конст.Значение.Substring(0, 9)

select спр;

return query2.SingleOrDefault();

}

}

[NotMapped]

public string ПериодСтр

{

get

{

var query = Константы1С.Константы.ЗапросДляПериодическогоЗначения(9697, ID, ДатаДляПериодическихРеквизитов);

var res = query.SingleOrDefault();

if (res == null)

return null;

return res.Значение.Substring(0, 10);

}

}

}

И соответственно сам запрос к периодически данным

public static System.Linq.IQueryable<Константы1С.ЗначениеПериодического> ЗапросДляПериодическогоЗначения(int ID, string OBJID, DateTime DATE)

{

var бд = ГлобальныйКонтекст.БД;

var query = (from Константа in бд.ТаблицаКонстанты

where Константа.ID == ID && Константа.OBJID == OBJID && Константа.DATE <= DATE

orderby Константа.DATE descending, Константа.TIME descending, Константа.DOCID descending, Константа.ROW\_ID descending

select new Константы1С.ЗначениеПериодического{ Значение = Константа.VALUE }).Take(1);

return query;

}

Буду рад если кому то это поможет.

Code Firstработает на MS SQL 2005 и выше. Это нужно учитывать, так как 1С 7.7 заточена под 2000

Для создания модели сначала запустите из 1С CodeFirstTo1C.ert

Для её работы нужно скачать 1CPP.dll отсюда

<http://www.1cpp.ru/index.php/File:Icpp-latest.rar#.D0.92.D0.B5.D1.80.D1.81.D0.B8.D1.8F_3.2.4.1>.

Пользуясь случаем хочу выразить огромную благодарность содателям этого чудного проекта

Установить НаименованиеБазы и namespace

В итоге будут показаны тексты модулей которые нужно будет скопировать в VS.

В VS создайте проект с именем которое указали в namespace.

В проекте выберите добавить класс, назовите его на ваше усмотрение и скопируйте из таблицы текст в него, предварительно переключив клавиатуру на RU.

После создания всех модулей перейдите в управление пакетами и установите

Entity Framework

Про строку подключения можно посмотреть здесь

<http://professorweb.ru/my/entity-framework/6/level2/2_9.php>

Но есть еще вариант посмотреть как реально работает Code First к существующей базе.

<http://metanit.com/sharp/entityframework/2.1.php>

Вы получите строку подключения и описание какой либо таблицы как я и делал на примере

И заменить в секции connectionStrings на своё имя