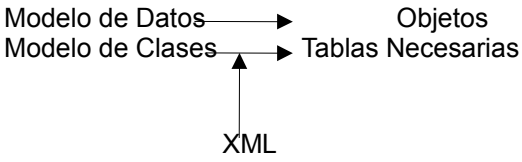
Apuntes Parcial 2 - ORM LINQ Nhibernate 2012

Seminario De Aplicación Profesional (Universidad Abierta Interamericana)

ORM

Es una técnica de programación para convertir datos entre el Lenguaje de Programación OO y el Sistema de Base de Datos relacional que estamos usando para nuestro desarrollo.

Mapeador de Objetos, se puede pensar desde el modulo de la Base de Datos (DER), y la herramienta la convierte en Objetos; esta forma no es la mejor, ya que no siempre corresponde una tabla con un Objeto. Es por eso, que generalmente se realiza de forma inversa, es decir, desde las Clases, se crean las clases que se necesitan. Para efectuar esto, utilizamos un XML para ayudar al ORM.



Ventajas ORM:

Más rápido en el Desarrollo (no hago el mapper a mano)
Abstracción de la DB
Reutilización
Seguridad (Limita el SQL Injection, ya que solo hablo con Objetos)
Mantenimiento de Código
Lenguaje propio de Consultas.

Desventajas ORM:

Tiempo utilizado en el aprendizaje: Este tipo de herramientas suelen ser complejas por lo que su correcta utilización lleva un tiempo que hay que emplear en ver el funcionamiento correcto y ver todo el partido que se le puede sacar.

Aplicaciones algo más lentas: Esto es debido a que todas las consultas que se hagan sobre la base de datos, el sistema primero deberá de transformarlas al lenguaje propio de la herramienta, luego leer los

Ejemplos de ORM:

Doctrine -- PHP
Propel -- PHP Trabaja directamente con Objetos
Hibernate
LinQ

LINQ (Language INtegrated Query)

Language-Integrated Query (LINQ) es una innovación introducida en Visual Studio 2008 que separa el mundo de los objetos y el mundo de los datos.
Integra conceptos de consultas directamente en los lenguajes de programación, permitiendo que el código de acceso a datos sea verificado por el compilador y las herramientas de desarrollador.
LINQ no impone a usar una arquitectura especifica más bien facilita la implementación de varias arquitecturas existentes para acceso a datos.

¿Porque LINQ?

Con frecuencia los programas deben acceder a diferentes dominios de datos:

Un documento XML
Una base de datos
Objetos en memoria

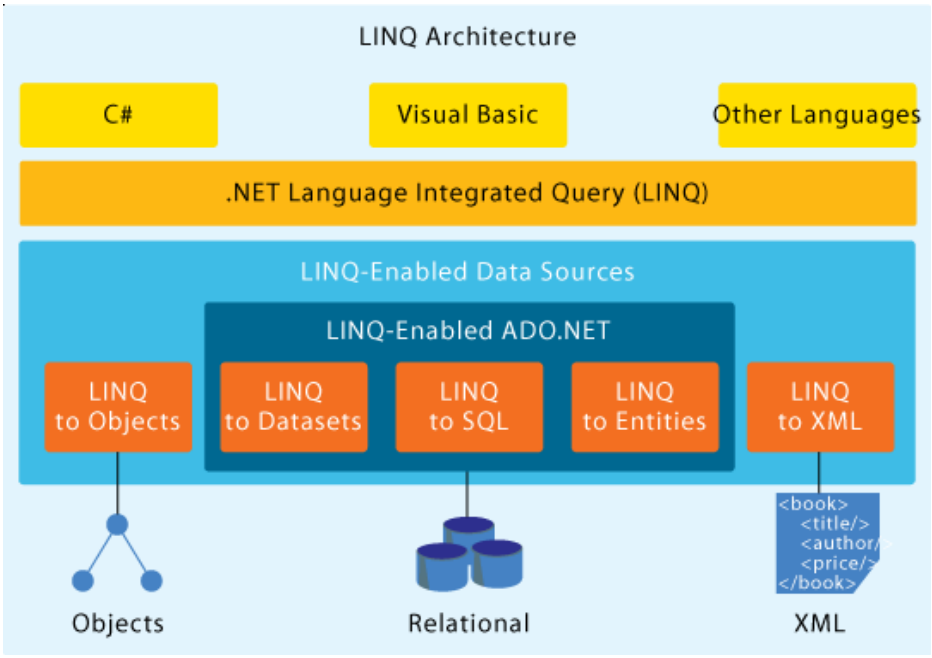
Cada dominio de datos tiene su propio modelo de acceso:
Bases de datos -> SQL
XML -> DOM , XQuery
Y entonces LINQ permite realizar consultas en todos estos con un solo lenguaje.

Ventajas Linq:

1. Capacidad de Fuentes Diversas:
    i. Objetos
    ii. XML
    iii. ADO .NET(DataSet, Entities, SQL

2. Misma sintaxis para accede a diferentes dominios de datos. Permite independizarse de los lenguajes de consultas específicos de cada uno.
3. Según el framework que uso, puedo usar la tecnología de LINQ que necesito. Ej, para Silverlight puedo usar LINQ para objetos y xml solamente.

Arquitectura:



Tecnología	.NET 3.5	.NET 3.5 CF	Silverlight 2.0	Ensamblado	Espacio de nombres
LINQ to Object	X	X	X	System.Core.dll	System.Linq
LINQ to XML	X	X	X	System.Xml.Linq.dll	System.Xml.Linq
LINQ to DataSet	X	X		System.Data.DataSetExtensions.dll	System.Data
LINQ to SQL	X			System.Data.Linq.dll	System.Data.Linq
LINQ to Entities	X			System.Data.Entity.dll	System.Data.Objects y otros

Operaciones LINQ

1. Obtener el Origen de datos
2. Crear Consulta
3. Ejecutar Consulta

La expresión básica de la consulta contiene tres clausulas:

From: Especifica el Origen
Where: Aplica el filtro
Select: Seleccionar datos

Aquí, lo importante es que, en LINQ, la propia variable de consulta no realiza ninguna acción ni devuelve datos. Simplemente almacena la información necesaria para generar los resultados cuando la consulta se ejecute posteriormente.

Adicional: GENERICS Una función recibe un valor como argumento sin saber que tipo es, sin embargo, cuando la uso, le paso el Argumento con el tipo

Operadores en LINQ

Restricción	Where
Proyección	Select, SelectMany
Ordenación	OrderBy, ThenBy
Agrupación	GroupBy
Encuentros	Join, GroupJoin
Cuantificadores	Any, All
Partición	Take, Skip, TakeWhile, SkipWhile
Conjuntuales	Distinct, Union, Intersect, Except
Un elemento	First, Last, Single, ElementAt
Agregados	Count, Sum, Min, Max, Average
Conversión	ToArray, ToList, ToDictionary
Conversión de elementos	OfType<T>, Cast<T>

Variantes:

Linq To Objects
Linq To XML
Linq to ADO NET
- Linq to DAtaSEt
- LinqToEntties
- LinqToSQL

LinqToObject:
Uso directo de consultas LINQ con cualquier colección IEnumerable o IEnumerable(Of T) sin utilizar ninguna API o proveedor LINQ intermedio, como LINQ to SQL o LINQ to XML. Puede utilizar LINQ para consultar cualquier colección enumerable,
como List(Of T), Array oDictionary(Of TKey, TValue).

IENUMERABLE <T> Esta interfaz la recorro con un FOREACH, puede devolver arrays, colecciones o listas. Utiliza el "tipo" T, que es un Generics

Ejemplo:

```
class IntroDeLINQ
{ static void Main() {
// Las 3 Partes de una consulta LINQ:
// 1. Origen de Datos.
Int = numeros = new int[7] { 0, 1, 2, 3, 4, 5, 6 };

// 2. Creación de la Consulta.
// numQuery es un IEnumerable<int>
var numQuery = from num in numeros where (num % 2) == 0 select num;

// 3. Ejecución de la consulta.
foreach (int num in numQuery) {Console.Write("{0,1} ", num);}}}
```

Linq To XL

Permite el acceso a datos XML usando la tecnología de LINQ
Puede consultar y modificar el documento
Permite escribir consultas en el documento XML en memoria para recuperar colecciones de elementos y atributos

Ejemplo:
Autos.xml

```
<?xml version='1.0'?>
<!-- Este documento XML representa un inventario de autos -->
<autos>
      <auto marca="BMW">
            <modelo>520</modelo>
            <potencia>125CV</potencia>
      </auto>
      <auto marca="BMW">
            <modelo>525</modelo>
            <potencia>135CV</potencia>
      </auto>
      <auto marca="Citroen">
            <modelo>C3</modelo>
            <potencia>75CV</potencia>
      </auto>
      <auto marca="Citroen">
```

```
            <modelo>C4</modelo>
            <potencia>115CV</potencia>
      </auto>
      <auto marca="Citroen">
            <modelo>C5</modelo>
            <potencia>135CV</potencia>
      </auto>
</autos>
```

```
public Form1() {
InitializeComponent(); }
private void button1_Click(object sender, EventArgs e)
{
XDocument documentoXML = XDocument.Load(@"C:\Autos.xml");
var datosSeleccionados = from datos in
documentoXML.Descendants("auto")
where datos.Attribute("marca").Value.ToString().ToUpper() == "BMW"
select new
{
modeloAuto = datos.Element("modelo").Value, potenciaAuto =
datos.Element("potencia").Value
};
string resultado = "";
foreach (var dato in datosSeleccionados)
{
resultado += String.Format("{0} tiene un auto {1} de {2} CV", "BMW",
dato.modeloAuto, dato.potenciaAuto) + "\r\n";
}
// Mostramos la información
MessageBox.Show(resultado); } }
```

LINQ to ADO .NET

Permite el acceso a datos ADO .NET usando la tecnología de LINQ
Se incluyen otras dos tecnologías
LINQ to DataSet
LINQ to SQL *

Linq To DataSet

Llenamos el DataSet y después construimos y ejecutamos consultas LINQ sobre esos datos en memoria

Linq To SQL

- SI queremos trabajar con fuente de datos conectadas
- LINQ to SQL permite la creación y manejo de la capa de acceso a datos contra una base de datos relacional
- Permite interactuar con SQL Server (solo soporta SQL Server y SQL Server Compact 3.5)
- Implementación de OR/M (mapeador de objetos relacionales)
- Modelar bases de datos relacionales con clase .NET
- Podemos consultar, actualizar, añadir, borrar
- LINQ to SQL convierte las consultas integradas en el lenguaje del modelo de objetos (C#) a SQL y las envía a la base de datos para su ejecución.

- Incluye compatibilidad con los procedimientos almacenados y las funciones definidas por el usuario en la base de datos
- Los chequeos se realizan en tiempo de compilación

Relacion de LINQ To SQL entre los elementos de un esquema de una BD

SQL	**LinqToSQL**
Base de Datos	Data Context
Tabla	Clase
Vista	Clase
Columna	Propiedad
Relacion	Propiedad
Stored Procedure	Método

Para poder utilizar LINQ to SQL se necesita:

- Acceso a SQL Server
- Un objeto **DataContext** (es el encargado de hacer la interface entre las entidades que son representadas y el programa.)
- Agregar las entidades a una clase o a un documento LINQ to SQL.

La Capa de Acceso a Datos se crea usando una de dos formas:
- Agregando un archivo LINQ to SQL (.dbml) al proyecto
- Generando un archivo de clases con la herramienta de generación (SqlMetal.exe)

Nhibernate

NHibernate es un framework de Object-Relational-Mapping open-source que resuelve en forma automática la persistencia de mis objetos de dominio .NET

Caracteristicas:
- Parte de las Clases
- Open Source
- Requiere XML para saber como debe persistir.
- Maneja sesiones
- Lo debo configurar en el Web.Config
- Tiene un file de Configuración

Pasos para el Uso de NHibernet

1. Crear la clase que necesita ser persistida
2. Crea la tabla para persistir la clase
3. Crear un archivo de mapeo (xml) para que Nhibernet sepa como persistir las propiedades de la clase
4. Crear un archivo de configuración para que NHIbernet sepa como conectarse a su BD (app.config o Web.config)
5. Usar la API de NHibernet

NHibernet vs LINQ 53

Con LINQ, la base de datos ya existe y las relaciones y un poco de programación dependerá de cómo se define la base de datos.

A diferencia de Linq, NHibernate es un código abierto.

NHibernate es una herramienta ORM, mientras que LINQ es una herramienta ORM incompleto, ya que las necesidades de extensiones adicionales.

Linq es ante todo un lenguaje de consultas, mientras que NHibernate tiene un lenguaje limitado de consultas.

LINQ es mucho más útil en aplicaciones pequeñas donde no hay dependencia masiva en las bases de datos.
NHibernate es muy flexible y ofrece más opciones.